Network Working Group
Request for Comments: 1190
Obsoletes: IEN-119

CIP Working Group
C. Topolcic, Editor
October 1990

Experimental Internet Stream Protocol, Version 2 (ST-II)

Status of this Memo

This memo defines a revised version of the Internet Stream Protocol, originally defined in IEN-119 [8], based on results from experiments with the original version, and subsequent requests, discussion, and suggestions for improvements. This is a Limited-Use Experimental Protocol. Please refer to the current edition of the "IAB Official Protocol Standards" for the standardization state and status of this protocol. Distribution of this memo is unlimited.

1. Abstract

This memo defines the Internet Stream Protocol, Version 2 (ST-II), an IP-layer protocol that provides end-to-end guaranteed service across an internet. This specification obsoletes IEN 119 "ST - A Proposed Internet Stream Protocol" written by Jim Forgie in 1979, the previous specification of ST. ST-II is not compatible with Version 1 of the protocol, but maintains much of the architecture and philosophy of that version. It is intended to fill in some of the areas left unaddressed, to make it easier to implement, and to support a wider range of applications.

1.1. Table of Contents

Status of this Memo 1
1. Abstract 1
1.1. Table of Contents 2
1.2. List of Figures 4

2. Introduction 7
2.1. Major Differences Between ST and ST-II 8
2.2. Concepts and Terminology 9
2.3. Relationship Between Applications and ST 11
2.4. ST Control Message Protocol 12
2.5. Flow Specifications 14

3. ST Control Message Protocol Functional Description . 17
3.1. Stream Setup 18
3.1.1. Initial Setup at the Origin 18
3.1.2. Invoking the Routing Function 19
3.1.3. Reserving Resources 19
3.1.4. Sending CONNECT Messages 20
3.1.5. CONNECT Processing by an Intermediate Agent . . 22
3.1.6. Setup at the Targets 23
3.1.7. ACCEPT Processing by an Intermediate Agent . . 24
3.1.8. ACCEPT Processing by the Origin 26
3.1.9. Processing a REFUSE Message 27
3.2. Data Transfer 30
3.3. Modifying an Existing Stream 31
3.3.1. Adding a Target 31
3.3.2. The Origin Removing a Target 33
3.3.3. A Target Deleting Itself 35
3.3.4. Changing the FlowSpec 36
3.4. Stream Tear Down 36
3.5. Exceptional Cases 37
3.5.1. Setup Failure due to CONNECT Timeout 37
3.5.2. Problems due to Routing Inconsistency 38
3.5.3. Setup Failure due to a Routing Failure . . . 39
3.5.4. Problems in Reserving Resources 41
3.5.5. Setup Failure due to ACCEPT Timeout 41
3.5.6. Problems Caused by CHANGE Messages 42
3.5.7. Notification of Changes Forced by Failures . . 42
3.6. Options 44
3.6.1. HID Field Option 44
3.6.2. PTP Option 44
3.6.3. FDx Option 45
3.6.4. NoRecovery Option 46
3.6.5. RevChrg Option 46
3.6.6. Source Route Option 46
3.7. Ancillary Functions 48
3.7.1. Failure Detection 48
3.7.1.1. Network Failures 48
3.7.1.2. Detecting ST Stream Failures 49
3.7.1.3. Subset 51

3.7.2. Failure Recovery 51
3.7.2.1. Subset 55
3.7.3. A Group of Streams 56
3.7.3.1. Group Name Generator 57
3.7.3.2. Subset 57
3.7.4. HID Negotiation 58
3.7.4.1. Subset 64
3.7.5. IP Encapsulation of ST 64
3.7.5.1. IP Multicasting 65
3.7.6. Retransmission 66
3.7.7. Routing 67
3.7.8. Security 67
3.8. ST Service Interfaces 68
3.8.1. Access to Routing Information 69
3.8.2. Access to Network Layer Resource Reservation . 70
3.8.3. Network Layer Services Utilized 71
3.8.4. IP Services Utilized 71
3.8.5. ST Layer Services Provided 72

4. ST Protocol Data Unit Descriptions 75
4.1. Data Packets 76
4.2. ST Control Message Protocol Descriptions 77
4.2.1. ST Control Messages 79
4.2.2. Common SCMP Elements 80
4.2.2.1. DetectorIPAddress 80
4.2.2.2. ErroredPDU 80
4.2.2.3. FlowSpec & RFlowSpec 81
4.2.2.4. FreeHIDs 84
4.2.2.5. Group & RGroup 85
4.2.2.6. HID & RHID 86
4.2.2.7. MulticastAddress 86
4.2.2.8. Name & RName 87
4.2.2.9. NextHopIPAddress 88
4.2.2.10. Origin 88
4.2.2.11. OriginTimestamp 89
4.2.2.12. ReasonCode 89
4.2.2.13. RecordRoute 94
4.2.2.14. SrcRoute 95
4.2.2.15. Target and TargetList 96
4.2.2.16. UserData 98
4.2.3. ST Control Message PDUs 99
4.2.3.1. ACCEPT 100
4.2.3.2. ACK 102
4.2.3.3. CHANGE-REQUEST 103
4.2.3.4. CHANGE 104
4.2.3.5. CONNECT 105
4.2.3.6. DISCONNECT 110
4.2.3.7. ERROR-IN-REQUEST 111
4.2.3.8. ERROR-IN-RESPONSE 112
4.2.3.9. HELLO 113
4.2.3.10. HID-APPROVE 114
4.2.3.11. HID-CHANGE-REQUEST 115

4.2.3.12. HID-CHANGE 116
4.2.3.13. HID-REJECT 118
4.2.3.14. NOTIFY 120
4.2.3.15. REFUSE 122
4.2.3.16. STATUS 124
4.2.3.17. STATUS-RESPONSE 126
4.3. Suggested Protocol Constants 127

5. Areas Not Addressed 131

6. Glossary 135

7. References 143

8. Security Considerations. 144

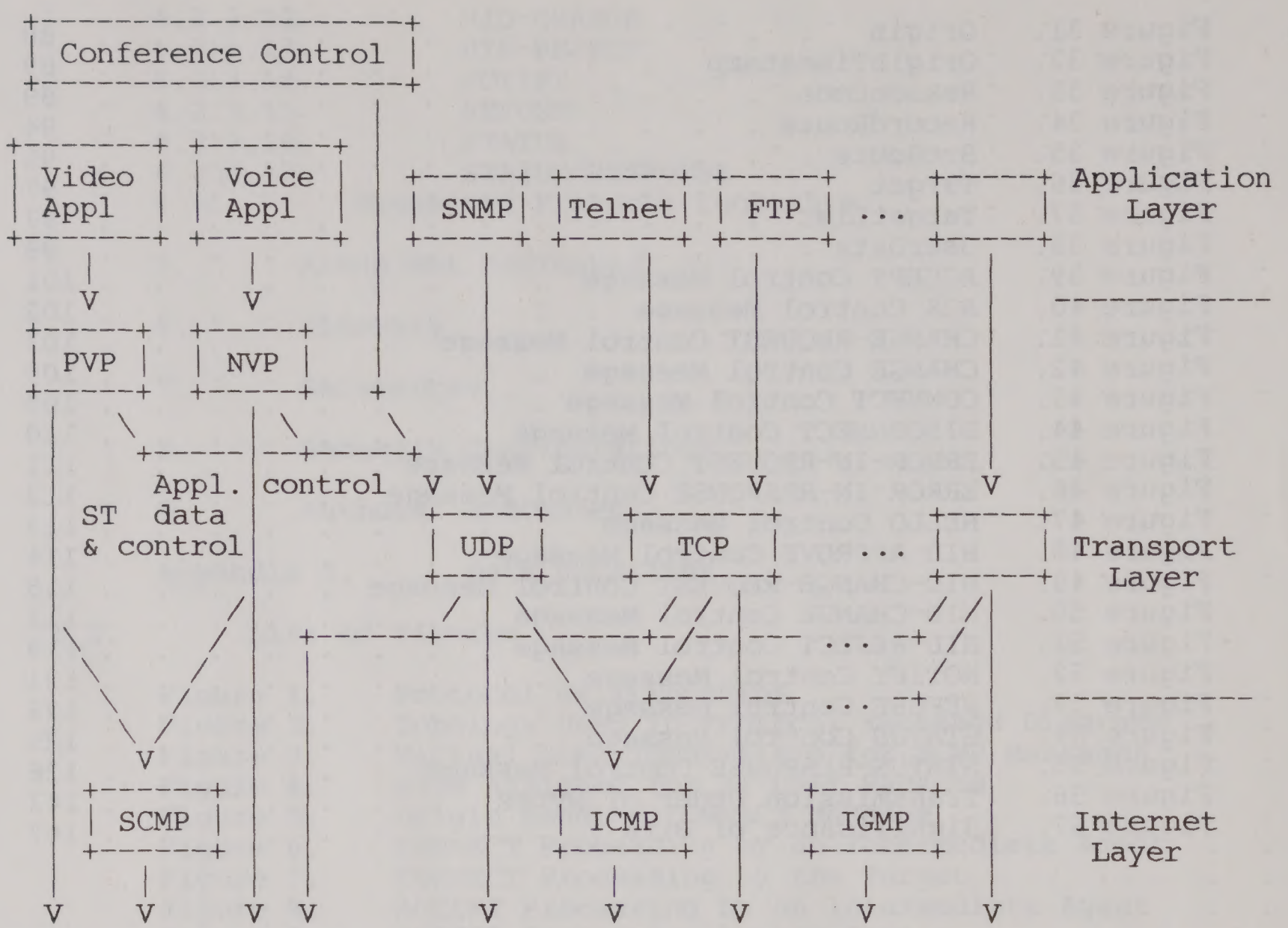
9. Authors' Addresses 145

Appendix 1. Data Notations 147

1.2. List of Figures

Figure 1. Protocol Relationships 6
Figure 2. Topology Used in Protocol Exchange Diagrams . . 16
Figure 3. Virtual Link Identifiers for SCMP Messages . . 16
Figure 4. HIDs Assigned for ST User Packets 18
Figure 5. Origin Sending CONNECT Message 21
Figure 6. CONNECT Processing by an Intermediate Agent . . 22
Figure 7. CONNECT Processing by the Target 24
Figure 8. ACCEPT Processing by an Intermediate Agent . . 25
Figure 9. ACCEPT Processing by the Origin 26
Figure 10. Sending REFUSE Message 28
Figure 11. Routing Around a Failure 29
Figure 12. Addition of Another Target 32
Figure 13. Origin Removing a Target 34
Figure 14. Target Deleting Itself 35
Figure 15. CONNECT Retransmission after a Timeout 38
Figure 16. Processing NOTIFY Messages 43
Figure 17. Source Routing Option 47
Figure 18. Typical HID Negotiation (No Multicasting) . . . 60
Figure 19. Multicast HID Negotiation 61
Figure 20. Multicast HID Re-Negotiation 62
Figure 21. ST Header 75
Figure 22. ST Control Message Format 77
Figure 23. ErroredPDU 80
Figure 24. FlowSpec & RFlowSpec 81
Figure 25. FreeHIDs 85
Figure 26. Group & RGroup 85
Figure 27. HID & RHID 86
Figure 28. MulticastAddress 86
Figure 29. Name & RName 87
Figure 30. NextHopIPAddress 88

Figure 31. Origin 88
Figure 32. OriginTimestamp 89
Figure 33. ReasonCode 89
Figure 34. RecordRoute 94
Figure 35. SrcRoute 95
Figure 36. Target 97
Figure 37. TargetList 97
Figure 38. UserData 98
Figure 39. ACCEPT Control Message 101
Figure 40. ACK Control Message 102
Figure 41. CHANGE-REQUEST Control Message 103
Figure 42. CHANGE Control Message 105
Figure 43. CONNECT Control Message 109
Figure 44. DISCONNECT Control Message 110
Figure 45. ERROR-IN-REQUEST Control Message 111
Figure 46. ERROR-IN-RESPONSE Control Message 112
Figure 47. HELLO Control Message 113
Figure 48. HID-APPROVE Control Message 114
Figure 49. HID-CHANGE-REQUEST Control Message 115
Figure 50. HID-CHANGE Control Message 117
Figure 51. HID-REJECT Control Message 119
Figure 52. NOTIFY Control Message 121
Figure 53. REFUSE Control Message 123
Figure 54. STATUS Control Message 125
Figure 55. STATUS-RESPONSE Control Message 126
Figure 56. Transmission Order of Bytes 147
Figure 57. Significance of Bits 147



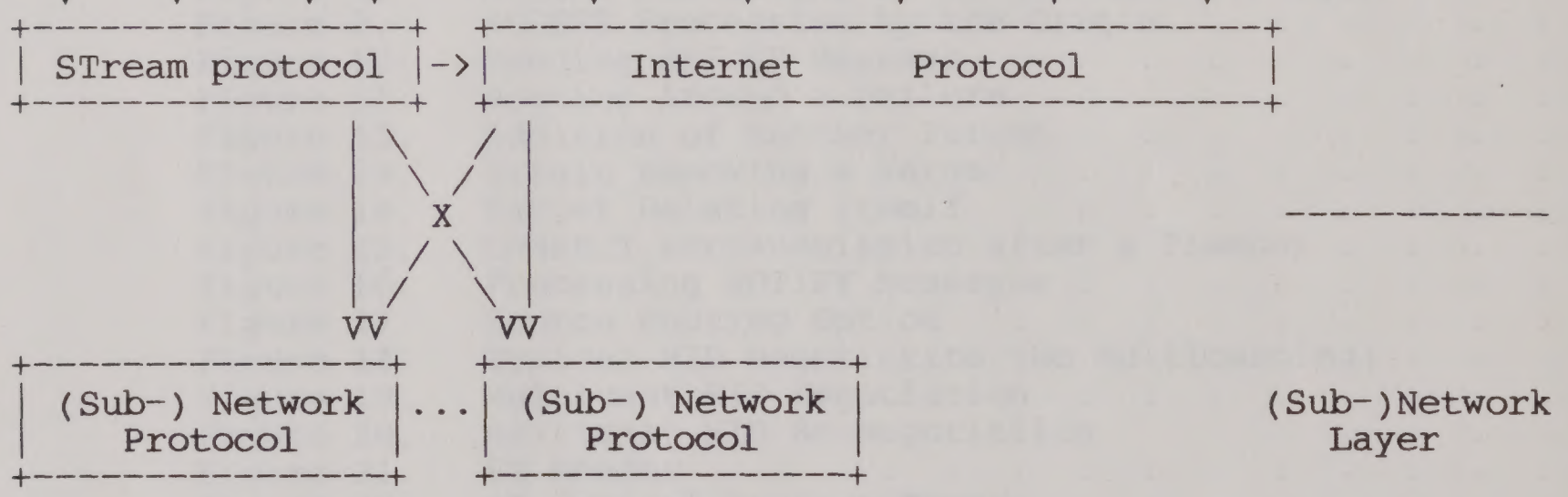


Figure 1. Protocol Relationships

2. Introduction

ST has been developed to support efficient delivery of streams of packets to either single or multiple destinations in applications requiring guaranteed data rates and controlled delay characteristics. The motivation for the original protocol was that IP [2] [15] did not provide the delay and data rate characteristics necessary to support voice applications.

ST is an internet protocol at the same layer as IP, see Figure 1. ST differs from IP in that IP, as originally envisioned, did not require routers (or intermediate systems) to maintain state information describing the streams of packets flowing through them. ST incorporates the concept of streams across an internet. Every intervening ST entity maintains state information for each stream that passes through it. The stream state includes forwarding information, including multicast support for efficiency, and resource information, which allows network or link bandwidth and queues to be assigned to a specific stream. This pre-allocation of resources allows data packets to be forwarded with low delay, low overhead, and a low probability of loss due to congestion. The characteristics of a stream, such as the number and location of the endpoints, and the bandwidth required, may be modified during the lifetime of the stream. This allows ST to give a real time application the guaranteed and predictable communication characteristics it requires, and is a good vehicle to support an application whose communications requirements are relatively predictable.

ST proved quite useful in several early experiments that involved voice conferences in the Internet. Since that time, ST has also been used to support point-to-point streams that include both video and voice. Recently, multimedia conferencing applications have been developed that need to exchange real-time voice, video, and pointer data in a multi-site conferencing environment. Multimedia conferencing across an internet is an application for which ST provides ideal support. Simulation and wargaming applications [14] also place similar requirements on the communication system. Other applications may include scientific visualization between a number of workstations and one or more remote supercomputers, and the collection and distribution of real-time sensor data from remote sensor platforms. ST may also be useful to support activities that are currently supported by IP, such as bulk file transfer using TCP.

Transport protocols above ST include the Packet Video Protocol (PVP) [5] and the Network Voice Protocol (NVP) [4], which are end-to-end protocols used directly by applications. Other transport layer protocols that may be used over ST include TCP [16], VMTP [3], etc. They provide the user interface, flow control, and packet ordering. This specification does not describe these higher layer protocols.

2.1. Major Differences Between ST and ST-II

ST-II supports a wider variety of applications than did the original ST. The differences between ST and ST-II are fairly straight forward yet provide great improvements. Four of the more notable differences are:

1 ST-II is decoupled from the Access Controller (AC). The AC, as well as providing a rudimentary access control function, also served as a centralized repository and distributor of the conference information. If an AC is necessary, it should be an entity in a higher layer protocol. A large variety of applications such as conferencing, distributed simulations, and wargaming can be run without an explicit AC.

2 The basic stream construct of ST-II is a directed tree carrying traffic away from a source to all the destinations, rather than the original ST's omniplex structure. For example, a conference is composed of a number of such trees, one for traffic from each participant. Although there are more (simplex) streams in ST-II, each is much simpler to manage, so the aggregate is much simpler. This change has a minimal impact on the application.

3 ST-II defines a number of the robustness and recovery mechanisms that were left undefined in the original ST specification. In case of a network or ST Agent failure, a stream may optionally be repaired automatically (i.e., without intervention from the user or the application) using a pruned depth first search starting at the ST Agent immediately preceding the failure.

4 ST-II does not make an inherent distinction between streams connecting only two communicants and streams among an arbitrary number of communicants.

This memo is the specification for the ST-II Protocol. Since there should be no ambiguity between the original ST specification and the specification herein, the protocol is simply called ST hereafter.

ST is the protocol used by ST entities to exchange information. The same protocol is used for communication among all ST entities, whether they communicate with a higher layer protocol or forward ST packets between attached networks.

The remainder of this section gives a brief overview of the ST Protocol. Section 3 (page 17) provides a detailed description of the operations required by the protocol. Section 4 (page 75) provides descriptions of the ST Protocol Data Units exchanged

between ST entities. Issues that have not yet been fully addressed are presented in Section 5 (page 131). A glossary and list of references are in Sections 6 (page 135) and 7 (page 143), respectively.

This memo also defines "subsets" of ST that can be implemented. A subsetted implementation does not have full ST functionality, but it can interoperate with other similarly subsetted implementations, or with a full implementation, in a predictable and consistent manner. This approach allows an implementation to be built and provide service with minimum effort, and gives it an immediate and well defined growth path.

2.2. Concepts and Terminology

The ST packet header is not constrained to be compatible with the IP packet header, except for the IP Version Number (the first four bits) that is used to distinguish ST packets (IP Version 5) from IP packets (IP Version 4). The ST packets, or protocol data units (PDUs), can be encapsulated in IP either to provide connectivity (possibly with degraded service) across portions of an internet that do not provide support for ST, or to allow access to services such as security that are not provided directly by ST.

An internet entity that implements the ST Protocol is called an "ST Agent". We refer to two kinds of ST agents: "host ST agents", also called "host agents" and "intermediate ST agents", also called "intermediate agents". The ST agents functioning as hosts are sourcing or sinking data to a higher layer protocol or application, while ST agents functioning as intermediate agents are forwarding data between directly attached networks. This distinction is not part of the protocol, but is used for conceptual purposes only. Indeed, a given ST agent may be simultaneously performing both host and intermediate roles. Every ST agent should be capable of delivering packets to a higher layer protocol. Every ST agent can replicate ST data packets as necessary for multi-destination delivery, and is able to send packets whether received from a network interface or a higher layer protocol. There are no other kinds of ST agents.

ST provides applications with an end-to-end flow oriented service across an internet. This service is implemented using objects called "streams". ST data packets are not considered to be totally independent as are IP data packets. They are transmitted only as part of a point-to-point or point-to-multi- point stream. ST creates a stream during a setup phase before data is transmitted. During the setup phase, routes are selected and internetwork resources are reserved. Except for explicit changes to the stream, the routes remain in effect until the stream is explicitly torn down.

An ST stream is:

- the set of paths that data generated by an application entity traverses on its way to its peer application entity(s) that receive it,
- the resources allocated to support that transmission of data, and
- the state information that is maintained describing that transmission of data.

Each stream is identified by a globally unique "Name"; see Section 4.2.2.8 (page 87). The Name is specified in ST control operations, but is not used in ST data packets. A set of streams may be related as members of a larger aggregate called a "group". A group is identified by a "Group Name"; see Section 3.7.3 (page 56).

The end-users of a stream are called the "participants" in the stream. Data travels in a single direction through any given stream. The host agent that transmits the data into the stream is called the "origin", and the host agents that receive the data are called the "targets". Thus, for any stream one participant is the origin and the others are the targets.

A stream is "multi-destination simplex" since data travels across it in only one direction: from the origin to the targets. A stream can be viewed as a directed tree in which the origin is the root, all the branches are directed away from the root toward the targets, which are the leaves. A "hop" is an edge of that tree. The ST agent that is on the end of an edge in the direction toward the origin is called the "previous-hop ST agent", or the "previous-hop". The ST agents that are one hop away from a previous-hop ST agent in the direction toward the targets are called the "next-hop ST agents", or the "next-hops". It is possible that multiple edges between a previous-hop and several next-hops are actually implemented by a network level multicast group.

Packets travel across a hop for one of two purposes: data or control. For ST data packet handling, hops are marked by "Hop IDentifiers" (HIDs) used for efficient forwarding instead of the stream's Name. A HID is negotiated among several agents so that data forwarding can be done efficiently on both a point-to-point and multicast basis. All control message exchange is done on a point-to-point basis between a pair of agents. For control message handling, Virtual Link Identifiers are used to quickly dispatch the control messages to the proper stream's state machine.

ST requires routing decisions to be made at several points in the stream setup and management process. ST assumes that an appropriate routing algorithm exists to which ST has access; see Section 3.8.1 (page 69). However, routing is considered to be a separate issue. Thus neither the routing algorithm nor its implementation is specified here. A routing algorithm may attempt to minimize the number of hops to the target(s), or it may be more intelligent and attempt to minimize the total internet resources consumed. ST operates equally well with any reasonable routing algorithm. The availability of a source routing option does not eliminate the need for an appropriate routing algorithm in ST agents.

2.3. Relationship Between Applications and ST

It is the responsibility of an ST application entity to exchange information among its peers, usually via IP, as necessary to determine the structure of the communication before establishing the ST stream. This includes:

- o identifying the participants,
- o determining which are targets for which origins,
- o selecting the characteristics of the data flow between any origin and its target(s),
- o specifying the protocol that resides above ST,
- o identifying the Service Access Point (SAP), port, or socket relevant to that protocol at every participant, and
- o ensuring security, if necessary.

The protocol layer above ST must pass such information down to the ST protocol layer when creating a stream.

ST uses a flow specification, abbreviated herein as "FlowSpec", to describe the required characteristics of a stream. Included are bandwidth, delay, and reliability parameters. Additional parameters may be included in the future in an extensible manner. The FlowSpec describes both the desired values and their minimal allowable values. The ST agents thus have some freedom in allocating their resources. The ST agents accumulate information that describes the characteristics of the chosen path and pass that information to the origin and the targets of the stream.

ST stream setup control messages carry some information that is not specifically relevant to ST, but is passed through the interface to the protocol that resides above ST. The "next

protocol identifier" ("NextPcol") allows ST to demultiplex streams to a number of possible higher layer protocols. The SAP associated with each participant allows the higher layer protocol to further demultiplex to a specific application entity. A UserData parameter is provided; see Section 4.2.2.16 (page 98).

2.4. ST Control Message Protocol

ST agents create and manage a stream using the ST Control Message Protocol (SCMP). Conceptually, SCMP resides immediately above ST (as does ICMP above IP) but is an integral part of ST. Control messages are used to:

- o create streams,
- o refuse creation of a stream,
- o delete a stream in whole or in part,
- o negotiate or change a stream's parameters,
- o tear down parts of streams as a result of router or network failures, or transient routing inconsistencies, and
- o reroute around network or component failures.

SCMP follows a request-response model. SCMP reliability is ensured through use of retransmission after timeout; see Section 3.7.6 (page 66).

An ST application that will transmit data requests its local ST agent, the origin, to create a stream. While only the origin requests creation of a stream, all the ST agents from the origin to the targets participate in its creation and management. Since a stream is simplex, each participant that wishes to transmit data must request that a stream be created.

An ST agent that receives an indication that a stream is being created must:

1. negotiate a HID with the previous-hop identifying the stream,
2. map the list of targets onto a set of next-hop ST agents through the routing function,
3. reserve the local and network resources required to support the stream,

4 update the FlowSpec, and

5 propagate the setup information and partitioned target list to the next-hop ST agents.

When a target receives the setup message, it must inquire from the specified application process whether or not it is willing to accept the stream, and inform the origin accordingly.

Once a stream is established, the origin can safely send data. ST and its implementations are optimized to allow fast and efficient forwarding of data packets by the ST agents using the HIDs, even at the cost of adding overhead to stream creation and management. Specifically, the forwarding decisions, that is, determining the set of next-hop ST agents to which a data packet belonging to a particular stream will be sent, are made during the stream setup phase. The shorthand HIDs are negotiated at that time, not only to reduce the data packet header size, but to access efficiently the stream's forwarding information. When possible, network-layer multicast is used to forward a data packet to multiple next-hop ST agents across a network. Note that when network-layer multicast is used, all members of the multicast group must participate in the negotiation of a common HID.

An established stream can be modified by adding or deleting targets, or by changing the network resources allocated to it. A stream may be torn down by either the origin or the targets. A target can remove itself from a stream leaving the others unaffected. The origin can similarly remove any subset of the targets from its stream leaving the remainder unaffected. An origin can also remove all the targets from the stream and eliminate the stream in its entirety.

A stream is monitored by the involved ST agents. If they detect a failure, they can attempt recovery. In general, this involves tearing down part of the stream and rebuilding it to bypass the failed component(s). The rebuilding always occurs from the origin side of the failure. The origin can optionally specify whether recovery is to be attempted automatically by intermediate ST agents or whether a failure should immediately be reported to the origin. If automatic recovery is selected but an intermediate agent determines it cannot effect the repair, it propagates the failure information backward until it reaches an agent that can effect repair. If the failure information propagates back to the origin, then the application can decide if it should abort or reattempt the recovery operation.

Although ST supports an arbitrary connection structure, we recognize that certain stream topologies will be common and justify special features, or options, which allow for optimized support. These include:

- o streams with only a single target (see Section 3.6.2 (page 44)), and
- o pairs of streams to support full duplex communication between two points (see Section 3.6.3 (page 45)).

These features allow the most frequently occurring topologies to be supported with less setup delay, with fewer control messages, and with less overhead than the more general situations.

2.5. Flow Specifications

Real time data, such as voice and video, have predictable characteristics and make specific demands of the networks that must transfer it. Specifically, the data may be transmitted in packets of a constant size that are produced at a constant rate. Alternatively, the bandwidth may vary, due either to variable packet size or rate, with a predefined maximum, and perhaps a non-zero minimum. The variation may also be predictable based on some model of how the data is generated. Depending on the equipment used to generate the data, the packet size and rate may be negotiable. Certain applications, such as voice, produce packets at the given rate only some of the time. The networks that support real time data must add minimal delay and delay variance, but it is expected that they will be non-zero.

The FlowSpec is used for three purposes. First, it is used in the setup message to specify the desired and minimal packet size and rate required by the origin. This information is used by ST agents when they attempt to reserve the resources in the intervening networks. Second, when the setup message reaches the target, the FlowSpec contains the packet size and rate that was actually obtained along the path from the origin, and the accrued mean delay and delay variance expected for data packets along that path. This information is used by the target to determine if it wishes to accept the connection. The target may reduce reserved resources if it wishes to do so and if the possibility is still available. Third, if the target accepts the connection, it returns the updated FlowSpec to the origin, so that the origin can decide if it still wishes to participate in the stream with the characteristics that were actually obtained.

When the data transmitted by stream users is generated at varying rates, including bursts of varying rate and duration, there is an opportunity to provide service to more subscribers by providing guaranteed service for the average data rate of each stream, and reserving additional network capacity, shared among all streams, to service the bursts. This concept has been recognized by analog voice network providers leading to the principle of time assigned speech interpolation (TASI) in which only the talkspurts of a speech conversation are transmitted, and, during silence periods, the circuit can be used to send the talkspurts of other conversations. The FlowSpec is intended to assist algorithms that perform similar kinds of functions. We do not propose such algorithms here, but rather expect that this will be an area for experimentation. To allow for experiments, and a range of ways that application traffic might be characterized, a "DutyFactor" is included in the FlowSpec and we expect that a "burst descriptor" will also be needed.

The FlowSpec will need to be revised as experience is gained with connections involving numerous participants using multiple media across heterogeneous internetworks. We feel a change of the FlowSpec does not necessarily require a new version of ST, it only requires the FlowSpec version number be updated and software to manage the new FlowSpec to be distributed. We further suggest that if the change to the FlowSpec involves additional information for improved operation, such as a burst descriptor, that it be added to the end of the FlowSpec and that the current parameters be maintained so that obsolete software can be used to process the current parameters with minimum modifications.

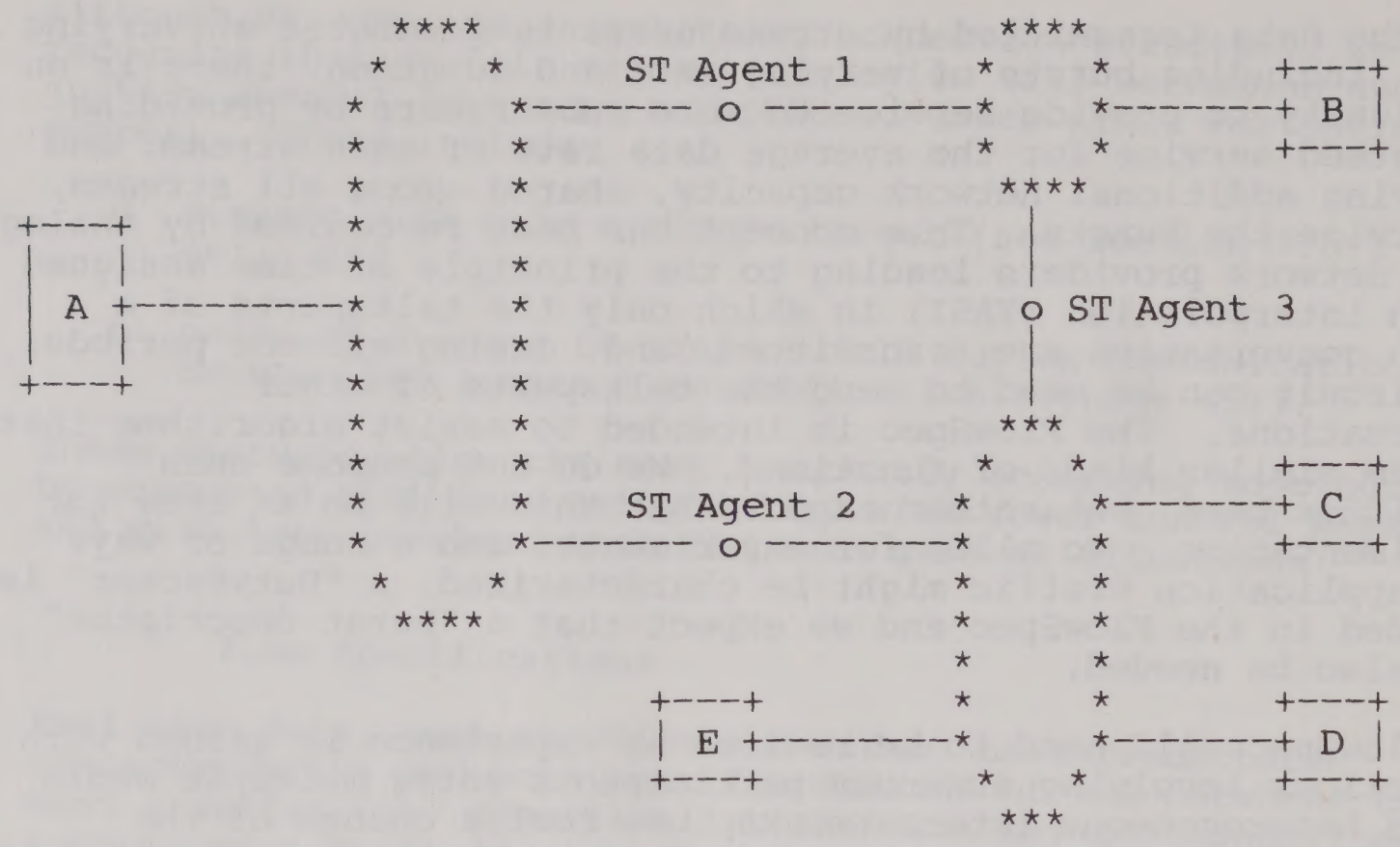


Figure 2. Topology Used in Protocol Exchange Diagrams

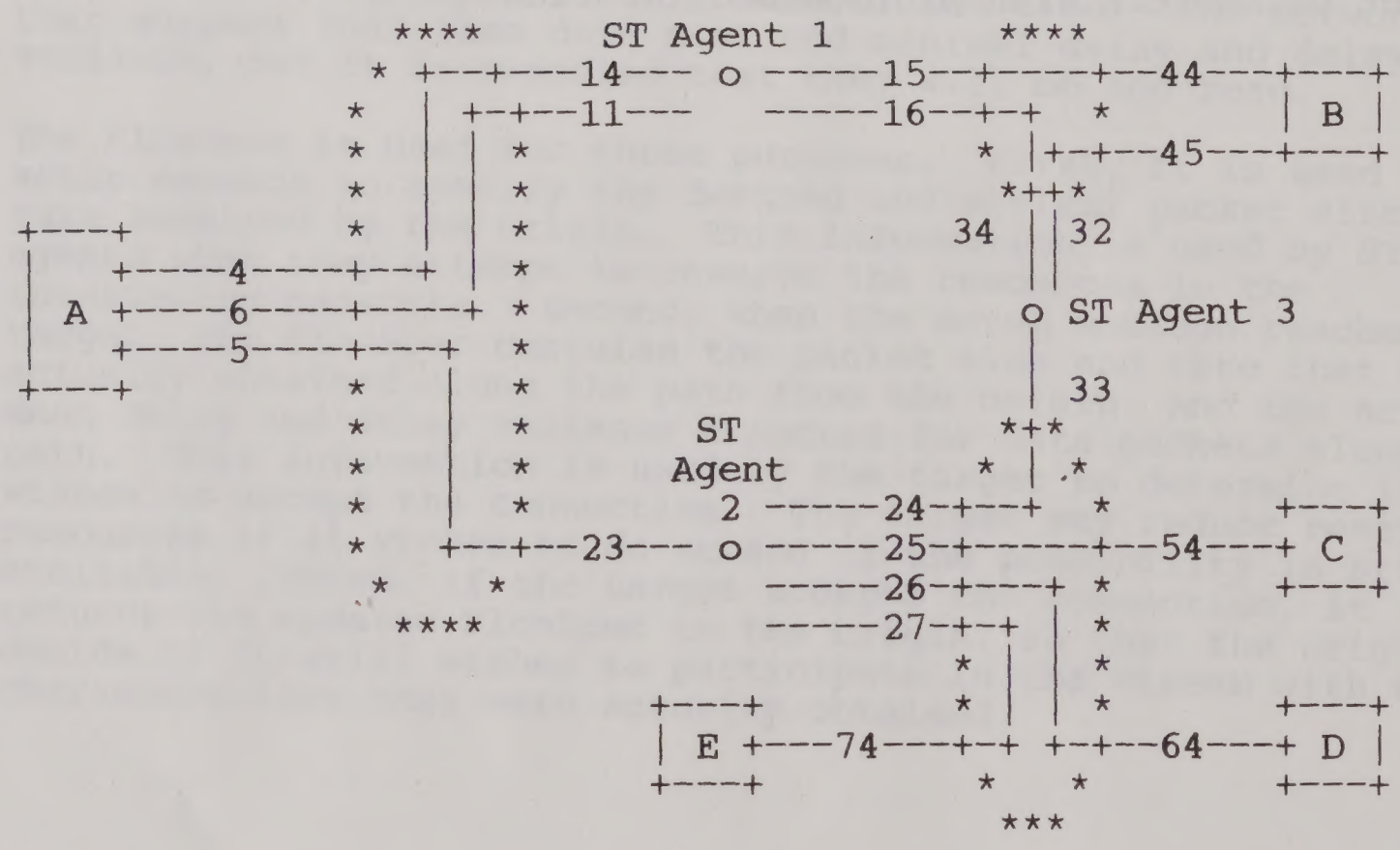


Figure 3. Virtual Link Identifiers for SCMP Messages

3. ST Control Message Protocol Functional Description

This section contains a functional description of the ST Control Message Protocol (SCMP); Section 4 (page 75) specifies the formats of the control message PDUs. We begin with a description of stream setup. Mechanisms used to deal with the exceptional cases are then presented. Complications due to options that an application or a ST agent may select are then detailed. Once a stream has been established, the data transfer phase is entered; it is described. Once the data transfer phase has been completed, the stream must be torn down and resources released; the control messages used to perform this function are presented. The resources or participants of a stream may be changed during the lifetime of the stream; the procedures to make changes are described. Finally, the section concludes with a description of some ancillary functions, such as failure detection and recovery, HID negotiation, routing, security, etc.

To help clarify the SCMP exchanges used to setup and maintain ST streams, we have included a series of figures in this section. The protocol interactions in the figures assume the topology shown in Figure 2. The figures, taken together,

- o Create a stream from an application at A to three peers at B, C and D,
- o Add a peer at E,
- o Disconnect peers B and C, and
- o D drops out of the stream.

Other figures illustrate exchanges related to failure recovery.

In order to make the dispatch function within SCMP more uniform and efficient, each end of a hop is assigned, by the agent at that end, a Virtual Link Identifier that uniquely (within that agent) identifies the hop and associates it with a particular stream's state machine(s). The identifier at the end of a link that is sending a message is called the Sender Virtual Link Identifier (SVLId); that at the receiving end is called the Receiver Virtual Link Identifier (RVLId). Whenever one agent sends a control message for the other to receive, the sender will place the receiver's identifier into the RVLId field of the message and its own identifier in the SVLId field. When a reply to the message is sent, the values in SVLId and RVLId fields will be reversed, reflecting the fact the sender and receiver roles are reversed. VLIds with values zero through three are received and should not be assigned in response to CONNECT messages. Figure 3 shows the hops that will be used in the examples and summarizes the VLIds that will be assigned to them.

Similarly, Figure 4 summarizes the HIDs that will eventually be negotiated as the stream is created.

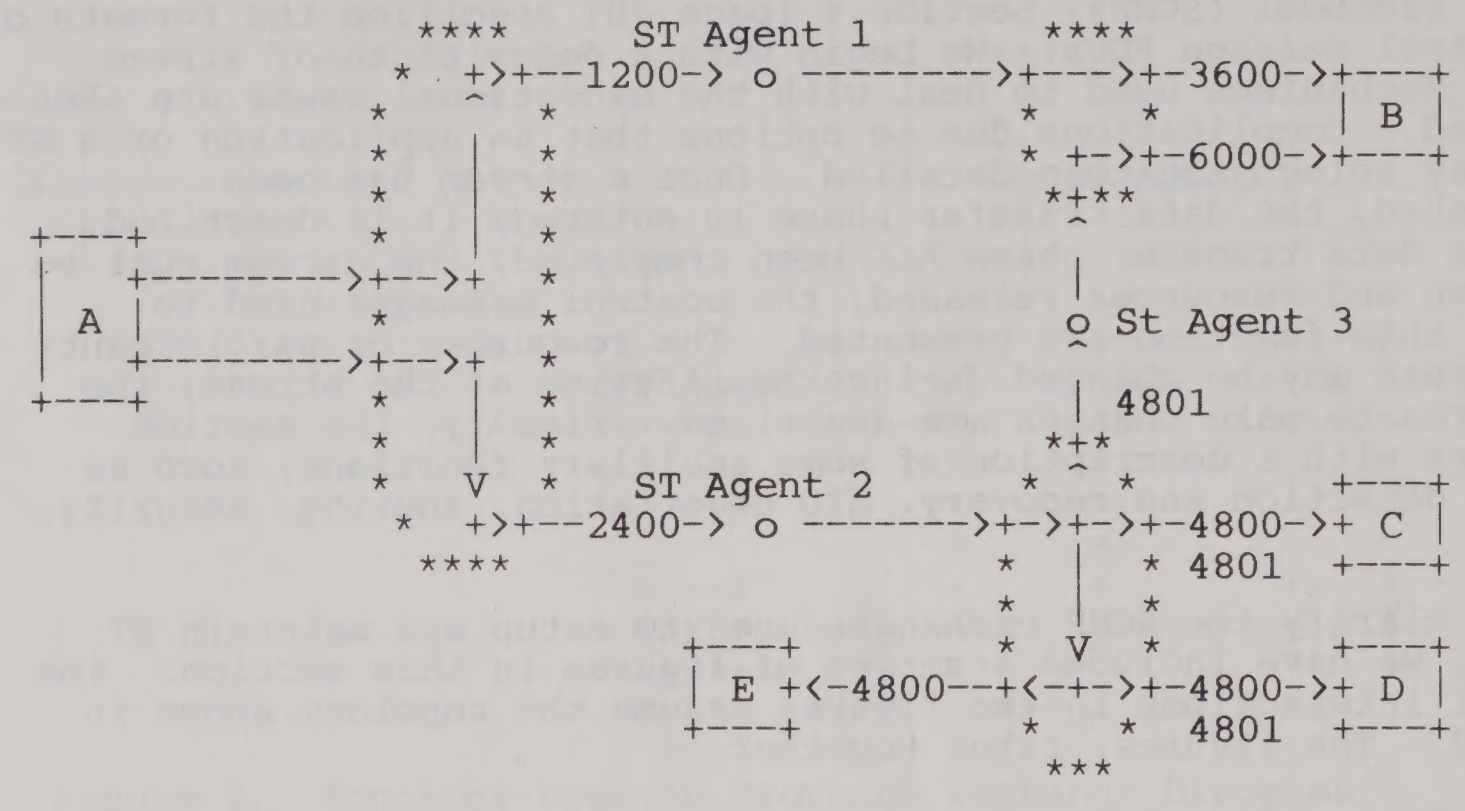


Figure 4. HIDs Assigned for ST User Packets

Some of the diagrams that follow form a progression. For example, the steps required initially to establish a connection are spread across five figures. Within a progression, the actions on the first diagram are numbered 1.1, 1.2, etc.; within the second diagram they are numbered 2.1, 2.2, etc. Points where control leaves one diagram to enter another are identified with a continuation arrow "-->>", and are continued with "[a.b] >>-->" in the other diagram. The number in brackets shows the label where control left the earlier diagram. The reception of simple acknowledgments, e.g., ACKs, in one figure from another is omitted for clarity.

3.1. Stream Setup

This section presents a description of stream setup assuming that everything succeeds -- HIDs are approved, any required resources are available, and the routing is correct.

3.1.1. Initial Setup at the Origin

As described in Section 2.3 (page 11), the application has collected the information necessary to determine the

participants in the communication before passing it to the host ST agent at the origin. The host ST agent will take this information, allocate a Name for the stream (see Section 4.2.2.8 (page 87)), and create a stream.

3.1.2. Invoking the Routing Function

An ST agent that is setting up a stream invokes a routing function to find a path to reach each of the targets specified in the TargetList. This is similar to the routing decision in IP. However, in this case the route is to a multitude of targets rather than to a single destination.

The set of next-hops that an ST agent would select is not necessarily the same as the set of next hops that IP would select given a number of independent IP datagrams to the same destinations. The routing algorithm may attempt to optimize parameters other than the number of hops that the packets will take, such as delay, local network bandwidth consumption, or total internet bandwidth consumption.

The result of the routing function is a set of next-hop ST agents and the parameters of the intervening network(s). The latter permit the ST agent to determine whether the selected network has the resources necessary to support the level of service requested in the FlowSpec.

3.1.3. Reserving Resources

The intent of ST is to provide a guaranteed level of service by reserving internet resources for a stream during a setup phase rather than on a per packet basis. The relevant resources are not only the forwarding information maintained by the ST agents, but also packet switch processor bandwidth and buffer space, and network bandwidth and multicast group identifiers. Reservation of these resources can help to increase the reliability and decrease the delay and delay variance with which data packets are delivered. The FlowSpec contains all the information needed by the ST agent to allocate the necessary resources. When and how these resources are allocated depends on the details of the networks involved, and is not specified here.

If an ST agent must send data across a network to a single next-hop ST agent, then only the point-to-point bandwidth needs to be reserved. If the agent must send data to multiple next-hop agents across one network and network layer multicasting is not available, then bandwidth must be reserved for all of them. This will allow the ST agent to

use replication to send a copy of the data packets to each next-hop agent.

If multicast is supported, its use will decrease the effort that the ST agent must expend when forwarding packets and also reduces the bandwidth required since one copy can be received by all next-hop agents. However, the setup phase is more complicated. A network multicast address must be allocated that contains all those next-hop agents, the sender must have access to that address, the next-hop agents must be informed of the address so they can join the multicast group identified by it (see Section 4.2.2.7 (page 86)), and a common HID must be negotiated.

The network should consider the bandwidth and multicast requirements to determine the amount of packet switch processing bandwidth and buffer space to reserve for the stream. In addition, the membership of a stream in a Group may affect the resources that have to be allocated; see Section 3.7.3 (page 56).

Few networks in the Internet currently offer resource reservation, and none that we know of offer reservation of all the resources specified here. Only the Terrestrial Wideband Network (TWBNet) [7] and the Atlantic Satellite Network (SATNET) [9] offer(ed) bandwidth reservation. Multicasting is more widely supported. No network provides for the reservation of packet switch processing bandwidth or buffer space. We hope that future networks will be designed to better support protocols like ST.

Effects similar to reservation of the necessary resources may be obtained even when the network cannot provide direct support for the reservation. Certainly if total reservations are a small fraction of the overall resources, such as packet switch processing bandwidth, buffer space, or network bandwidth, then the desired performance can be honored if the degree of confidence is consistent with the requirements as stated in the FlowSpec. Other solutions can be designed for specific networks.

3.1.4. Sending CONNECT Messages

A VLId and a proposed HID must be selected for each next-hop agent. The control packets for the next-hop must carry the VLId in the SVLId field. The data packets transmitted in the stream to the next-hop must carry the HID in the ST Header.

The ST agent sends a CONNECT message to each of the ST agents identified by the routing function. Each CONNECT message contains the VLId, the proposed HID (the HID Field option bit

must be set, see Section 3.6.1 (page 44)), an updated FlowSpec, and a TargetList. In general, the HID, FlowSpec, and TargetList will depend on both the next-hop and the intervening network. Each TargetList is a subset of the received (or original) TargetList, identifying the targets that are to be reached through the next-hop to which the CONNECT message is being sent. Note that a CONNECT message to a single next-hop might have to be fragmented into multiple CONNECTs if the single CONNECT is too large for the intervening network's MTU; fragmentation is performed by further dividing the TargetList.

If multiple next-hops are to be reached through a network that supports network level multicast, a different CONNECT message must nevertheless be sent to each next-hop since each will have a different TargetList; see Section 4.2.3.5 (page 105). However, since an identical copy of each ensuing data packet will reach each member of the multicast group, all the CONNECT messages must propose the same HID. See Section 3.7.4 (page 58) for a detailed discussion on HID selection.

In the example of Figure 2, the routing function might return that B is reachable via Agent 1 and C and D are reachable via Agent 2. Thus A would create two CONNECT messages, one each for Agents 1 and 2, as illustrated in Figure 5. Assuming that the proposed HIDs are available in the receiving agents, they would each send a responding HID-APPROVE back to Agent A.

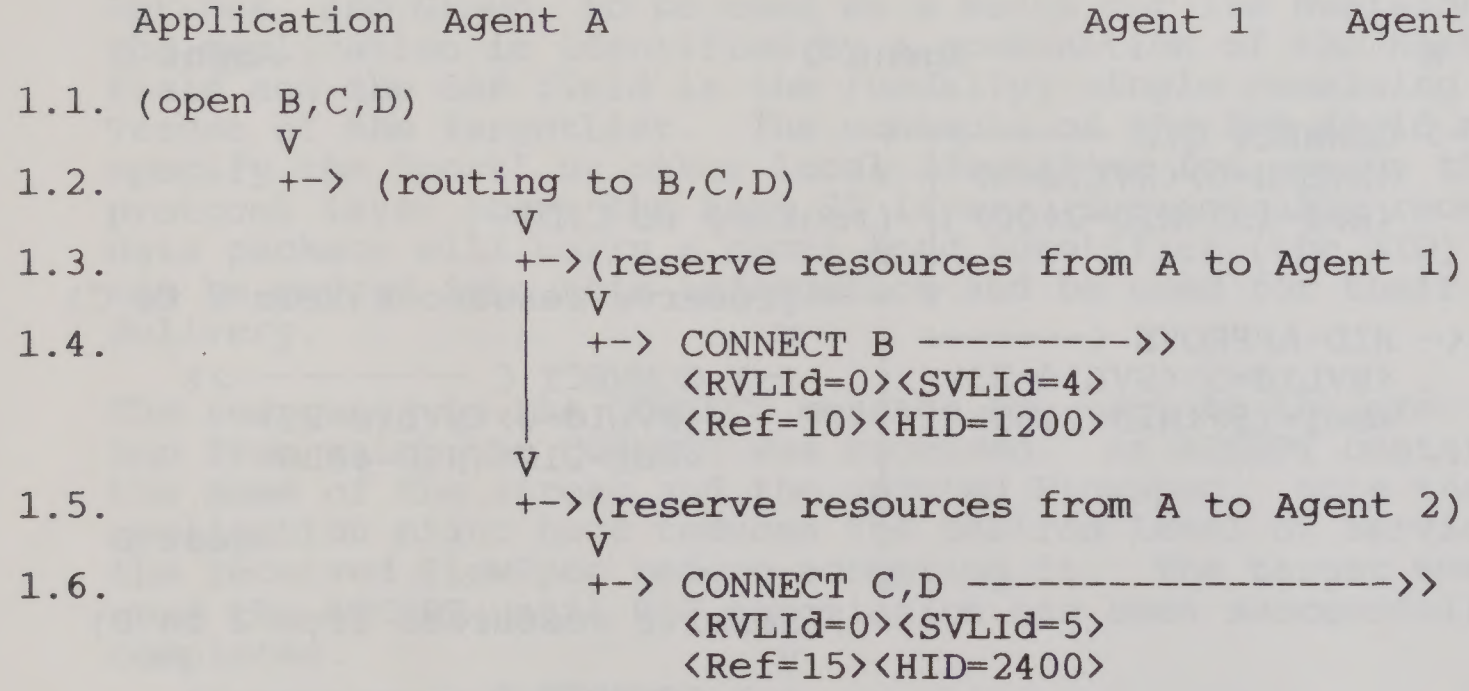


Figure 5. Origin Sending CONNECT Message

3.1.5. CONNECT Processing by an Intermediate Agent

An ST agent receiving a CONNECT message should,.assuming no errors, quickly select a VLId and respond to the previous-hop with either an ACK, a HID-REJECT, or a HID-APPROVE message, as is appropriate. This message must identify the CONNECT to which it corresponds by including the CONNECT's Reference number in its Reference field. Note that the VLId that this agent selects is placed in the SVLId of the response, and the previous-hop's VLId (which is contained in the SVLId of the CONNECT) is copied into the RVLId of the response. If the agent is not a target, it must then invoke the routing function, reserve resources, and send a CONNECT message(s) to its next-hop(s), as described in Sections 3.1.2-4 (pages 19-20).

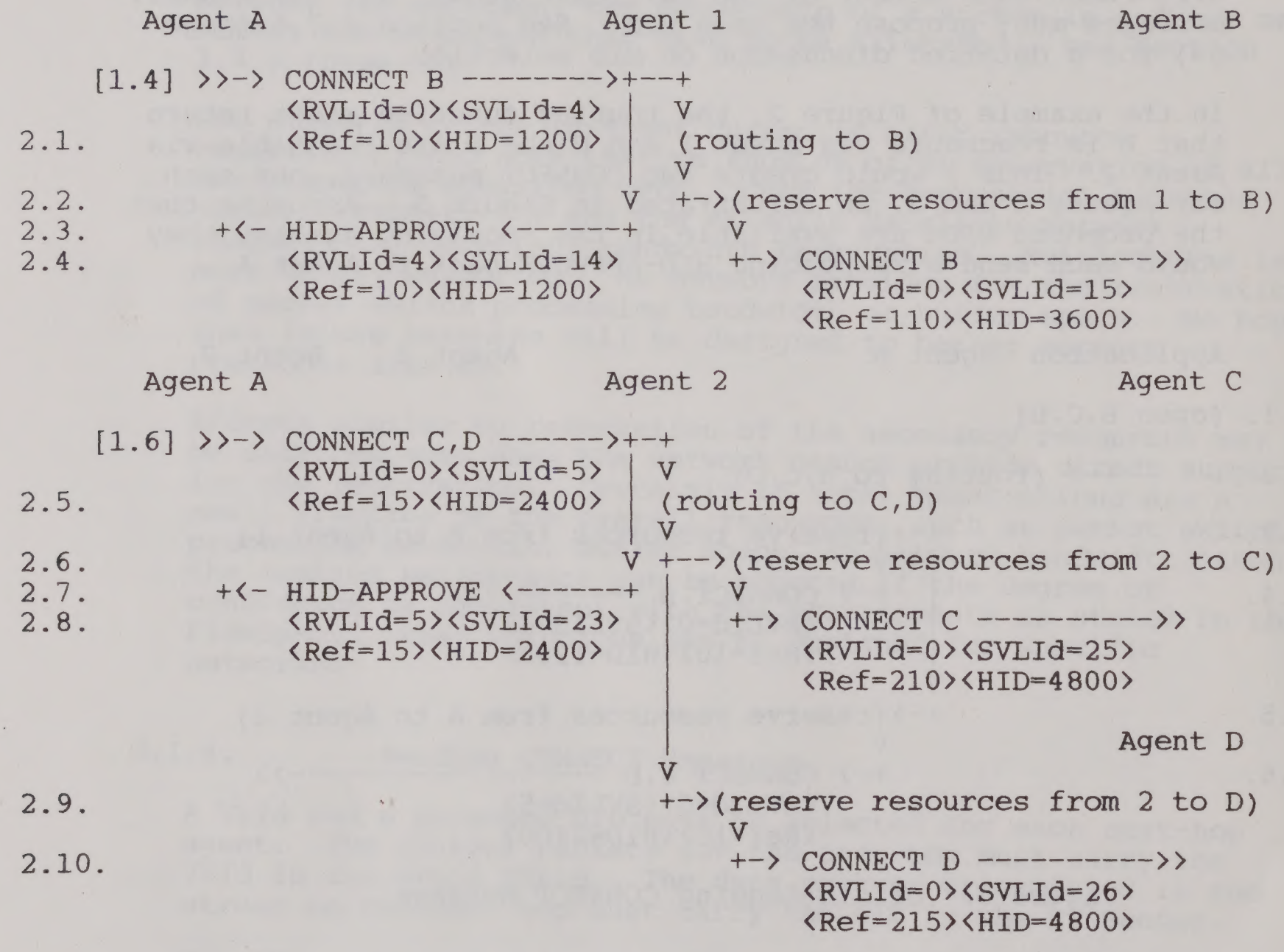


Figure 6. CONNECT Processing by an Intermediate Agent

The resources listed as Desired in a received FlowSpec may not correspond to those actually reserved in either the ST agent itself or in the network(s) used to reach the next-hop agent(s). As long as the reserved resources are sufficient to meet the specified Limits, the copy of the FlowSpec sent to a next-hop must have the Desired resources updated to reflect the resources that were actually obtained. For example, the Desired bandwidth might be reduced because the network to the next-hop could not provide all of the desired bandwidth. Also, the delay and delay variance are appropriately increased, and the link MTU may require that the DesPDUBytes field be reduced. (The minimum requirements that the origin had entered into the FlowSpec Limits fields cannot be altered by the intermediate or target agents.)

3.1.6. Setup at the Targets

An ST agent that is the target of a CONNECT, whether from an intermediate ST agent, or directly from the origin host ST agent, must respond first (assuming no errors) with either a HID-REJECT or HID-APPROVE. After inquiring from the specified application process whether or not it is willing to accept the connection, the agent must also respond with either an ACCEPT or a REFUSE.

In particular, the application must be presented with parameters from the CONNECT, such as the Name, FlowSpec, Options, and Group, to be used as a basis for its decision. The application is identified by a combination of the NextPcol field and the SAP field in the (usually) single remaining Target of the TargetList. The contents of the SAP field may specify the "port" or other local identifier for use by the protocol layer above the host ST layer. Subsequently received data packets will carry a short hand identifier (the HID) that can be mapped into this information and be used for their delivery.

The responses to the CONNECT message are sent to the previous-hop from which the CONNECT was received. An ACCEPT contains the Name of the stream and the updated FlowSpec. Note that the application might have reduced the desired level of service in the received FlowSpec before accepting it. The target must not send the ACCEPT until HID negotiation has been successfully completed.

Since the ACCEPT or REFUSE message must be acknowledged by the previous-hop, it is assigned a new Reference number that will be returned in the ACK. The CONNECT to which the ACCEPT or REFUSE is a reply is identified by placing the CONNECT's Reference number in the LnkReference field of the ACCEPT or REFUSE.

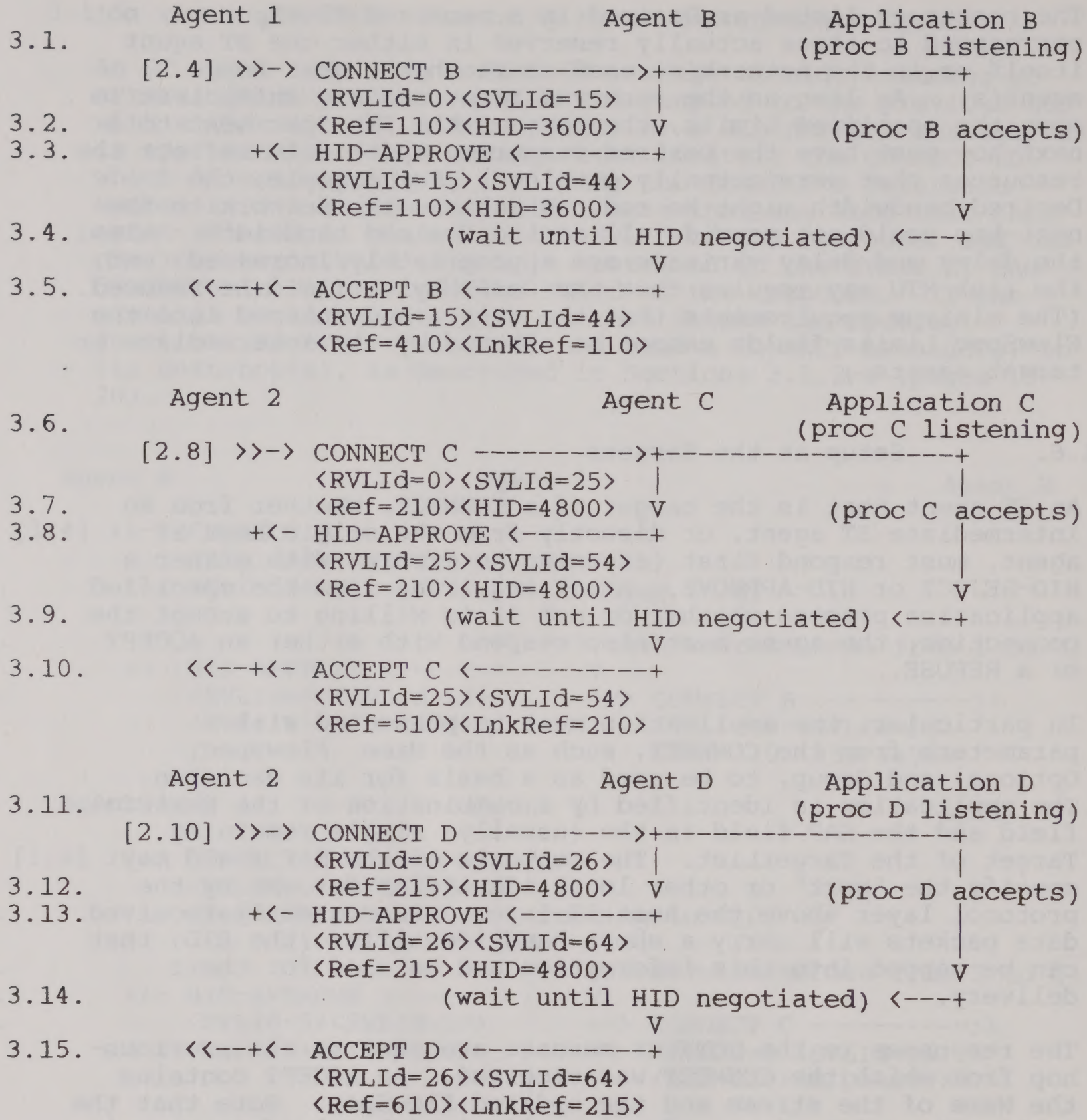


Figure 7. CONNECT Processing by the Target

3.1.7. ACCEPT Processing by an Intermediate Agent

When an intermediate ST agent receives an ACCEPT, it first verifies that the message is a response to an earlier CONNECT. If not, it responds to the next-hop ST agent with an ERROR-IN-REPLY (LnkRefUnknown) message. Otherwise, it responds to the next-hop ST agent with an ACK, and propagates

the ACCEPT message to the previous-hop along the same path traced by the CONNECT but in the reverse direction toward the origin. The ACCEPT should not be propagated until all HID negotiations with the next-hop agent(s) have been successfully completed.

The FlowSpec is included in the ACCEPT message so that the origin and intermediate ST agents can gain access to the information that was accumulated as the CONNECT traversed the internet. Note that the resources, as specified in the FlowSpec in the ACCEPT message, may differ from the resources that were reserved by the agent when the CONNECT was

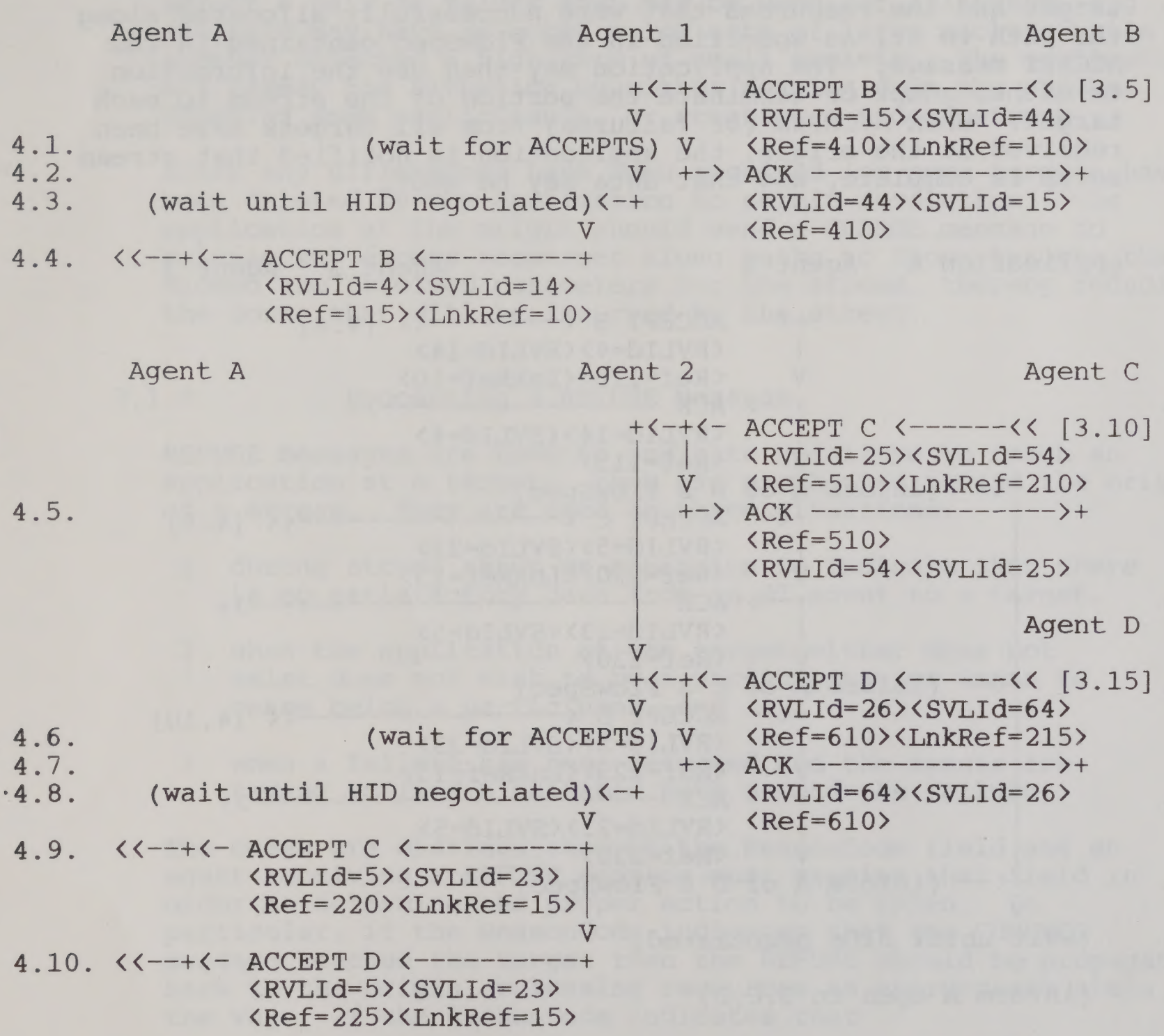


Figure 8. ACCEPT Processing by an Intermediate Agent

originally processed. However, the agent does not adjust the reservation in response to the ACCEPT. It is expected that any excess resource allocation will be released for use by other stream or datagram traffic through an explicit CHANGE message initiated by the application at the origin if it does not wish to be charged for any excess resource allocations.

3.1.8. ACCEPT Processing by the Origin

The origin will eventually receive an ACCEPT (or REFUSE or ERROR-IN-REQUEST) message from each of the targets. As each ACCEPT is received, the application should be notified of the target and the resources that were successfully allocated along the path to it, as specified in the FlowSpec contained in the ACCEPT message. The application may then use the information to either adopt or terminate the portion of the stream to each target. When ACCEPTs (or failures) from all targets have been received at the origin, the application is notified that stream setup is complete, and that data may be sent.

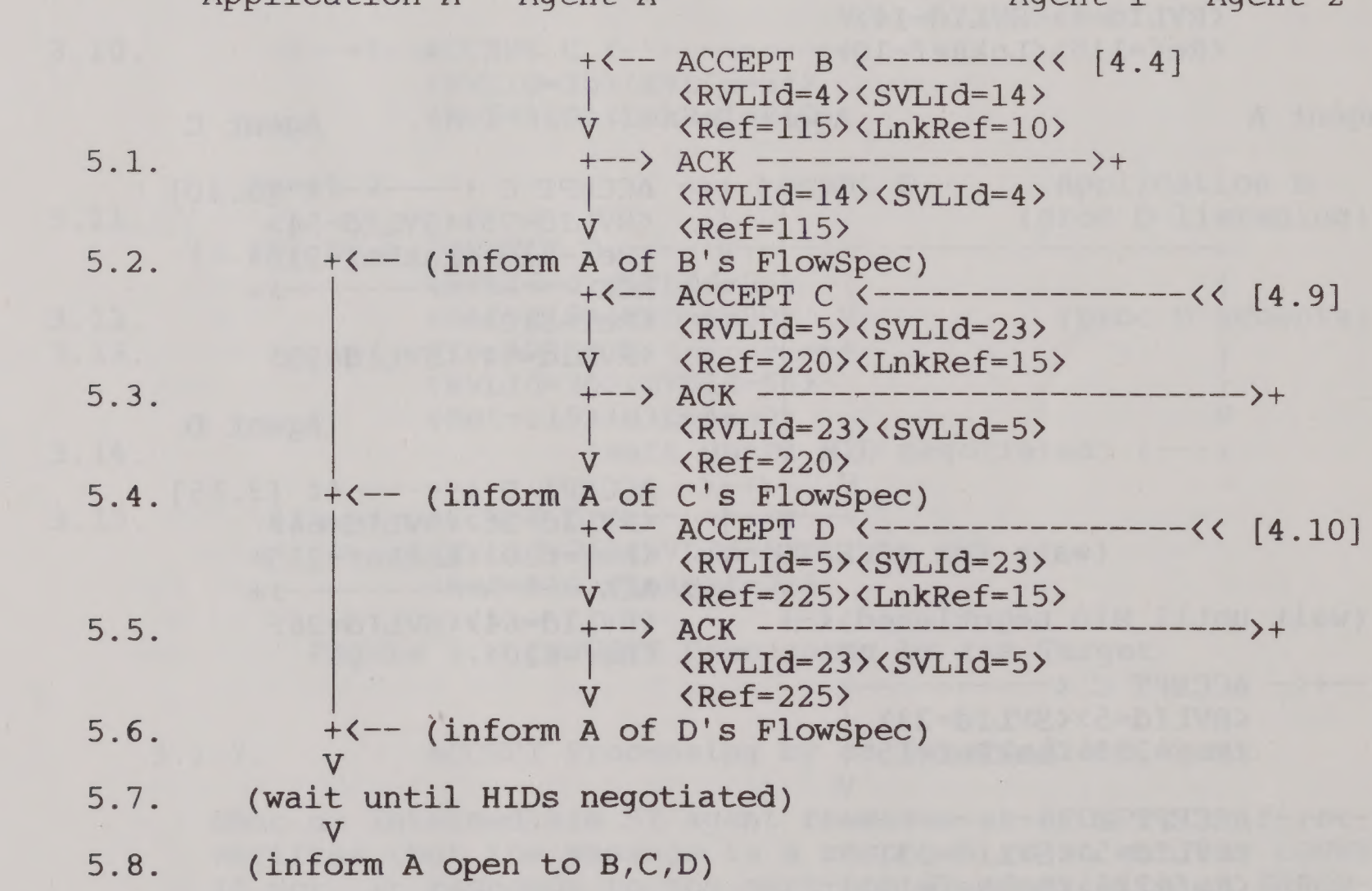


Figure 9. ACCEPT Processing by the Origin

There are several pieces of information contained in the FlowSpec that the application must combine before sending data through the stream. The PDU size should be computed from the minimum value of the DesPDUBytes field from all ACCEPTs and the protocol layers above ST should be informed of the limit. It is expected that the next higher protocol layer above ST will segment its PDUs accordingly. Note, however, that the MTU may decrease over the life of the stream if new targets are subsequently added. Whether the MTU should be increased as targets are dropped from a stream is left for further study.

The available bandwidth and packet rate limits must also be combined. In this case, however, it may not be possible to select a pair of values that may be used for all paths, e.g., one path may have selected a low rate of large packets while another selected a high rate of small packets. The application may remedy the situation by either tearing down the stream, dropping some participants, or creating a second stream.

After any differences have been resolved (or some targets have been deleted by the application to permit resolution), the application at the origin should send a CHANGE message to release any excess resources along paths to those targets that exceed the resolved parameters for the stream, thereby reducing the costs that will be incurred by the stream.

3.1.9. Processing a REFUSE Message

REFUSE messages are used to indicate a failure to reach an application at a target; they are propagated toward the origin of a stream. They are used in three situations:

1 during stream setup or expansion to indicate that there is no satisfactory path from an ST agent to a target,

2 when the application at the target either does not exist does not wish to be a participant, or wants to cease being a participant, and

3 when a failure has been detected and the agents are trying to find a suitable path around the failure.

The cases are distinguished by the ReasonCode field and an agent receiving a REFUSE message must examine that field in order to determine the proper action to be taken. In particular, if the ReasonCode indicates that the CONNECT message reached the target then the REFUSE should be propagated back to the origin, releasing resources as appropriate along the way. If the ReasonCode indicates that

the CONNECT message did not reach the target then the intermediate (origin) ST agent(s) should check for alternate routes to the target before propagating the REFUSE back another hop toward the origin. This implies that an agent must keep track of the next-hops that it has tried, on a target by target basis, in order not to get caught in a loop.

An ST agent that receives a REFUSE message must acknowledge it by sending an ACK to the next-hop. The REFUSE must also be propagated back to the previous-hop ST agent. Note that the ST agent may not have any information about the target in

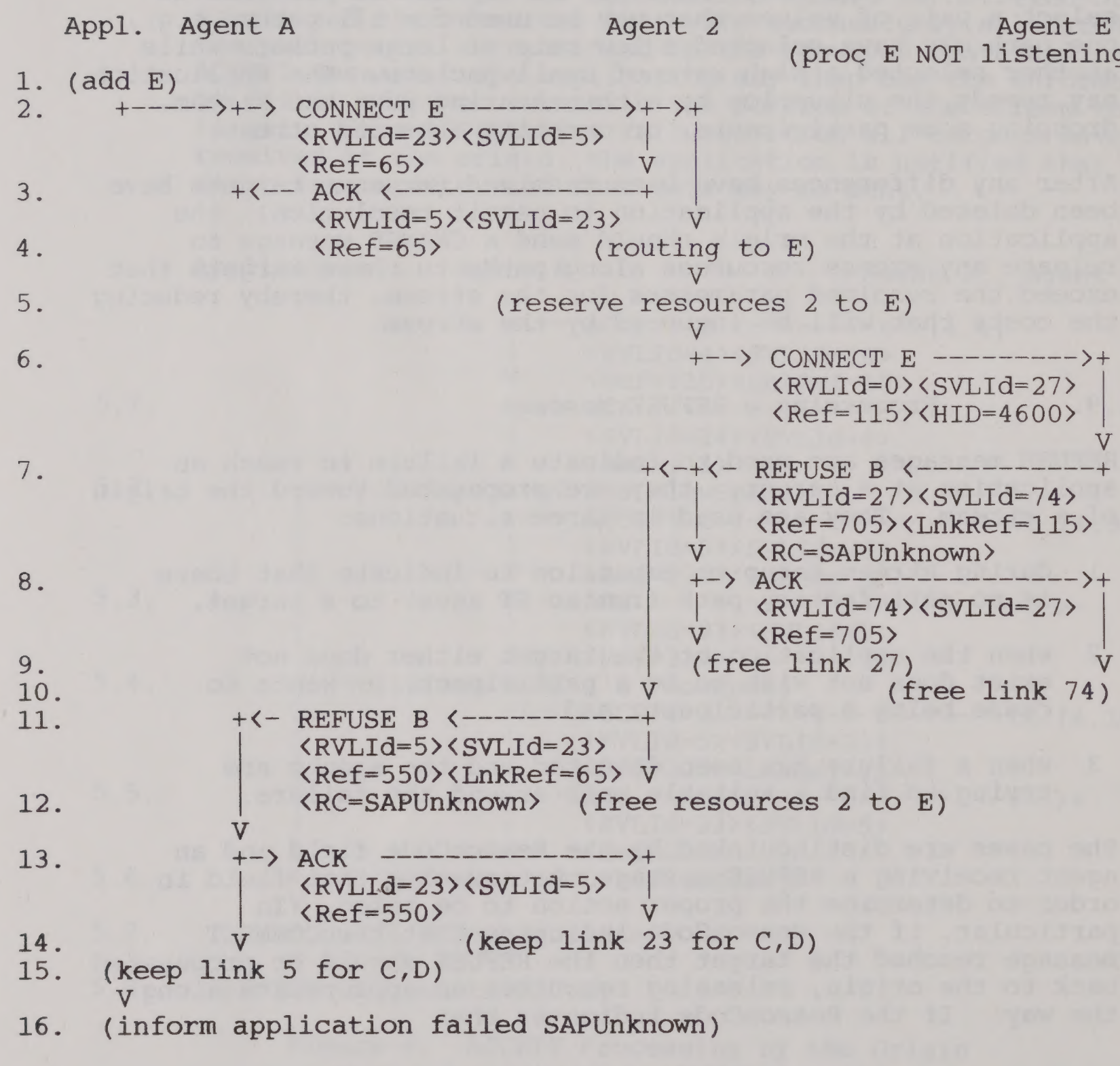


Figure 10. Sending REFUSE Message

the TargetList. This may result from interacting DISCONNECT and REFUSE messages and should be logged and silently ignored.

If, after deleting the specified target, the next-hop has no remaining targets, then those resources associated with that next-hop agent may be released. Note that network resources may not actually be released if network multicasting is being

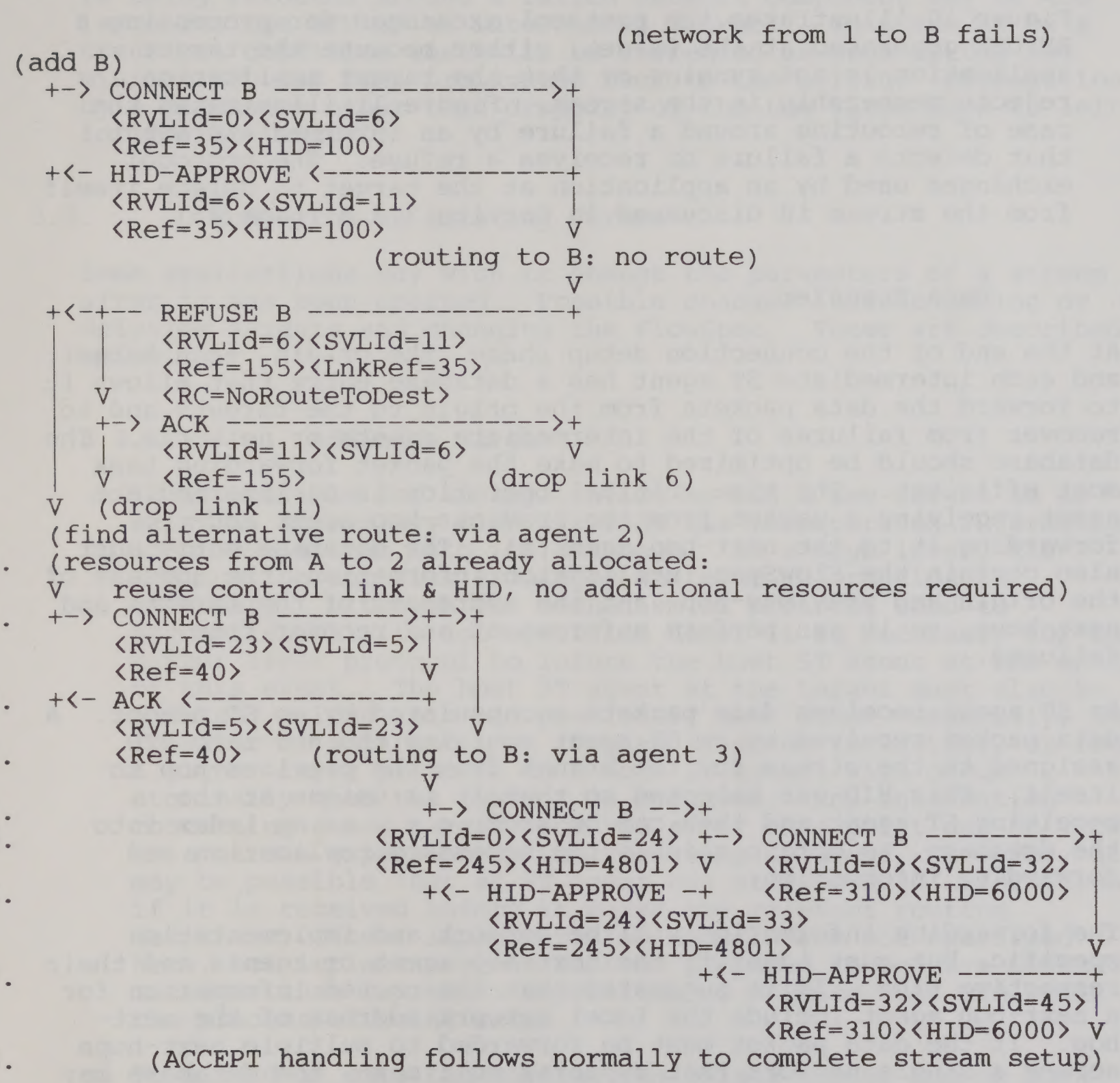


Figure 11. Routing Around a Failure

used since they may still be required for traffic to other next-hops in the multicast group.

When the REFUSE reaches a origin, the origin sends an ACK and notifies the application via the next higher layer protocol that the target listed in the TargetList is no longer part of the stream and also if the stream has no remaining targets. If there are no remaining targets, the application may wish to terminate the stream.

Figure 10 illustrates the protocol exchanges for processing a REFUSE generated at the target, either because the target application is not running or that the target application rejects membership in the stream. Figure 11 illustrates the case of rerouting around a failure by an intermediate agent that detects a failure or receives a refuse. The protocol exchanges used by an application at the target to delete itself from the stream is discussed in Section 3.3.3 (page 35).

3.2. Data Transfer

At the end of the connection setup phase, the origin, each target, and each intermediate ST agent has a database entry that allows it to forward the data packets from the origin to the targets and to recover from failures of the intermediate agents or networks. The database should be optimized to make the packet forwarding task most efficient. The time critical operation is an intermediate agent receiving a packet from the previous-hop agent and forwarding it to the next-hop agent(s). The database entry must also contain the FlowSpec, utilization information, the address of the origin and previous-hop, and the addresses of the targets and next-hops, so it can perform enforcement and recover from failures.

An ST agent receives data packets encapsulated by an ST header. A data packet received by an ST agent contains the non-zero HID assigned to the stream for the branch from the previous-hop to itself. This HID was selected so that it is unique at the receiving ST agent and thus can be used, e.g., as an index into the database, to obtain quickly the necessary replication and forwarding information.

The forwarding information will be network and implementation specific, but must identify the next-hop agent or agents and their respective HIDs. It is suggested that the cached information for a next-hop agent include the local network address of the next-hop. If the data packet must be forwarded to multiple next-hops across a single network that supports multicast, the database may specify a single HID and may identify the next-hops by a (local network) multicast address.

If the network does not support multicast, or the next-hops are on different networks, then the database must indicate multiple (next-hop, HID) tuples. When multiple copies of the data packet must be sent, it may be necessary to invoke a packet replicator.

Data packets should not require fragmentation as the next higher protocol layer at the origin was informed of the minimum MTU over all paths in the stream and is expected to segment its PDUs accordingly. However, it may be the case that a data packet that is being rerouted around a failed network component may be too large for the MTU of an intervening network. This should be a transient condition that will be corrected as soon as the new minimum MTU has been propagated back to the origin. Disposition by a mechanism other than dropping of the too large PDUs is left for further study.

3.3. Modifying an Existing Stream

Some applications may wish to change the parameters of a stream after it has been created. Possible changes include adding or deleting targets and changing the FlowSpec. These are described below.

3.3.1. Adding a Target

It is possible for an application to add a new target to an existing stream any time after ST has incorporated information about the stream into its database. At a high level, the application entities exchanges whatever information is necessary. Although the mechanism or protocol used to accomplish this is not specified here, it is necessary for the higher layer protocol to inform the host ST agent at the origin of this event. The host ST agent at the target must also be informed unless this had previously been done. Generally, the transfer of a target list from an ST agent to another, or from a higher layer protocol to a host ST agent, will occur atomically when the CONNECT is received. Any information concerning a new target received after this point can be viewed as a stream expansion by the receiving ST agent. However, it may be possible that an ST agent can utilize such information if it is received before it makes the relevant routing decisions. These implementation details are not specified here, but implementations must be prepared to receive CONNECT messages that represent expansions of streams that are still in the process of being setup.

To expand an existing stream, the origin issues one or more CONNECT messages that contain the Name, the VLId, the FlowSpec, and the TargetList specifying the new target or targets. The origin issues multiple CONNECT messages if

either the targets are to be reached through different next-hop agents, or a single CONNECT message is too large for the network MTU. The HID Field option is not set since the HID has already been (or is being) negotiated for the hop; consequently, the CONNECT is acknowledged with an ACK instead of a HID-REJECT or HID-APPROVE.

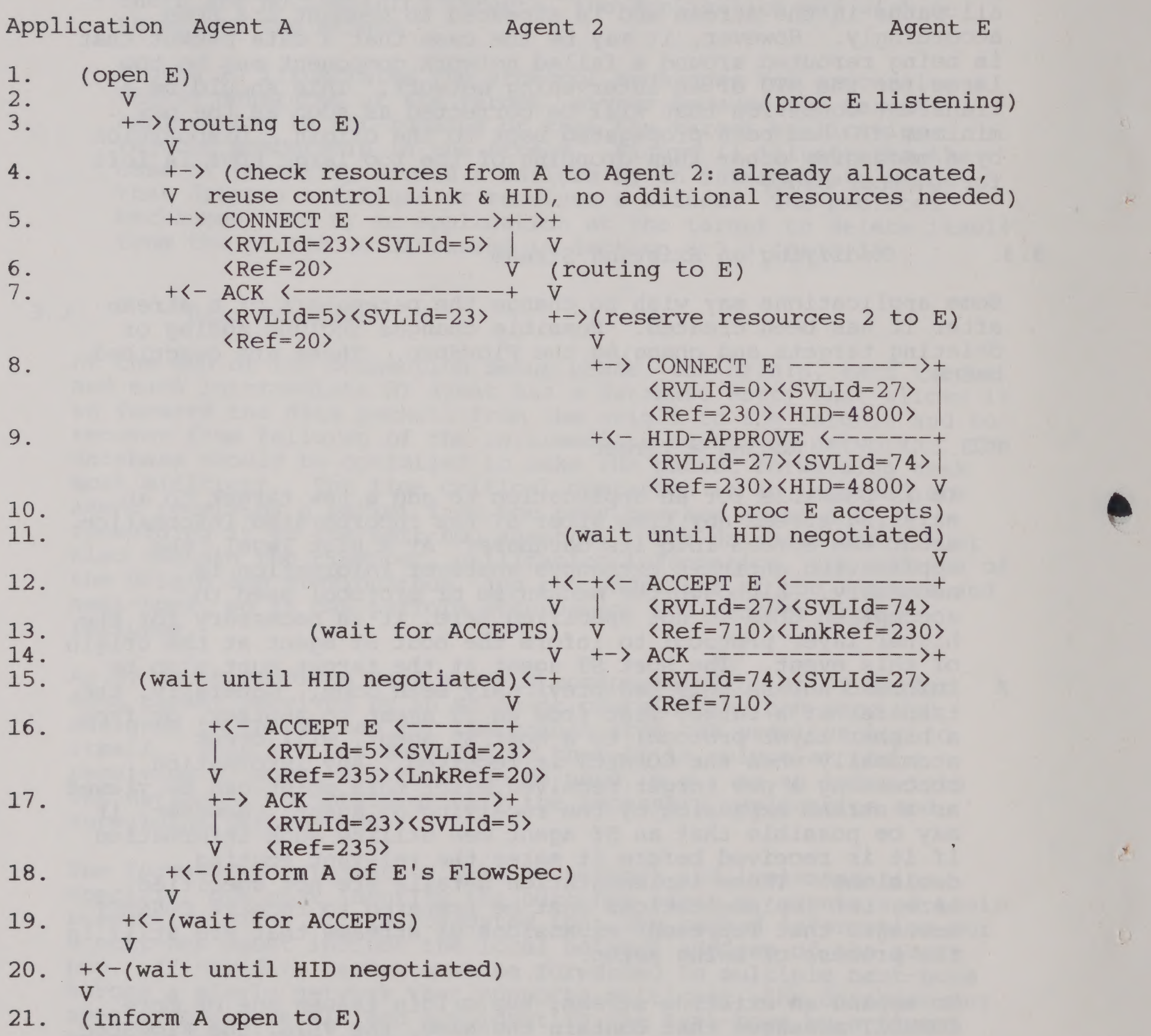


Figure 12. Addition of Another Target

An ST agent that is already a node in the stream recognizes the RVLId and verifies that the Name of the stream is the same. It then checks if the intersection of the TargetList and the targets of the established stream is empty. If this is not the case, then the receiver responds with an ERROR-IN-REQUEST with the appropriate reason code (RouteLoop) that contains a TargetList of those targets that were duplicates; see Section 4.2.3.5 (page 106).

For each new target in the TargetList, processing is much the same as for the original CONNECT; see Sections 3.1.2-4 (pages 19-20). The CONNECT must be acknowledged, propagated, and network resources must be reserved. However, it may be possible to route to the new targets using previously allocated paths or an existing multicast group. In that case, additional resources do not need to be reserved but more next-hop(s) might have to be added to an existing multicast group.

Nevertheless, the origin, or any intermediate ST agent that receives a CONNECT for an existing stream, can make a routing decision that is independent of any it may have made previously. Depending on the routing algorithm that is used, the ST agent may decide to reach the new target by way of an established branch, or it may decide to create a new branch. The fact that a new target is being added to an existing stream may result in a suboptimal overall routing for certain routing algorithms. We take this problem to be unavoidable since it is unlikely that the stream routing can be made optimal in general, and the only way to avoid this loss of optimality is to redefine the routing of potentially the entire stream, which would be too expensive and time consuming.

3.3.2. The Origin Removing a Target

The application at the origin specifies a set of targets that are to be removed from the stream and an appropriate reason code (ApplDisconnect). The targets are partitioned into multiple DISCONNECT messages based on the next-hop to the individual targets. As with CONNECT messages, an ST agent that is sending a DISCONNECT must make sure that the message fits into the MTU for the intervening network. If the message is too large, the TargetList must be further partitioned into multiple DISCONNECT messages.

An ST agent that receives a DISCONNECT message must acknowledge it by sending an ACK back to the previous-hop. The DISCONNECT must also be propagated to the relevant next-hop ST agents. Before propagating the message, however, the TargetList should be partitioned based on next-hop ST

agent and MTU, as described above. Note that there may be targets in the TargetList for which the ST agent has no information. This may result from interacting DISCONNECT and REFUSE messages and should be logged and silently ignored.

If, after deleting the specified targets, any next-hop has no remaining targets, then those resources associated with that next-hop agent may be released. Note that network resources may not actually be released if network multicasting is being used since they may still be required for traffic to other next-hops in the multicast group.

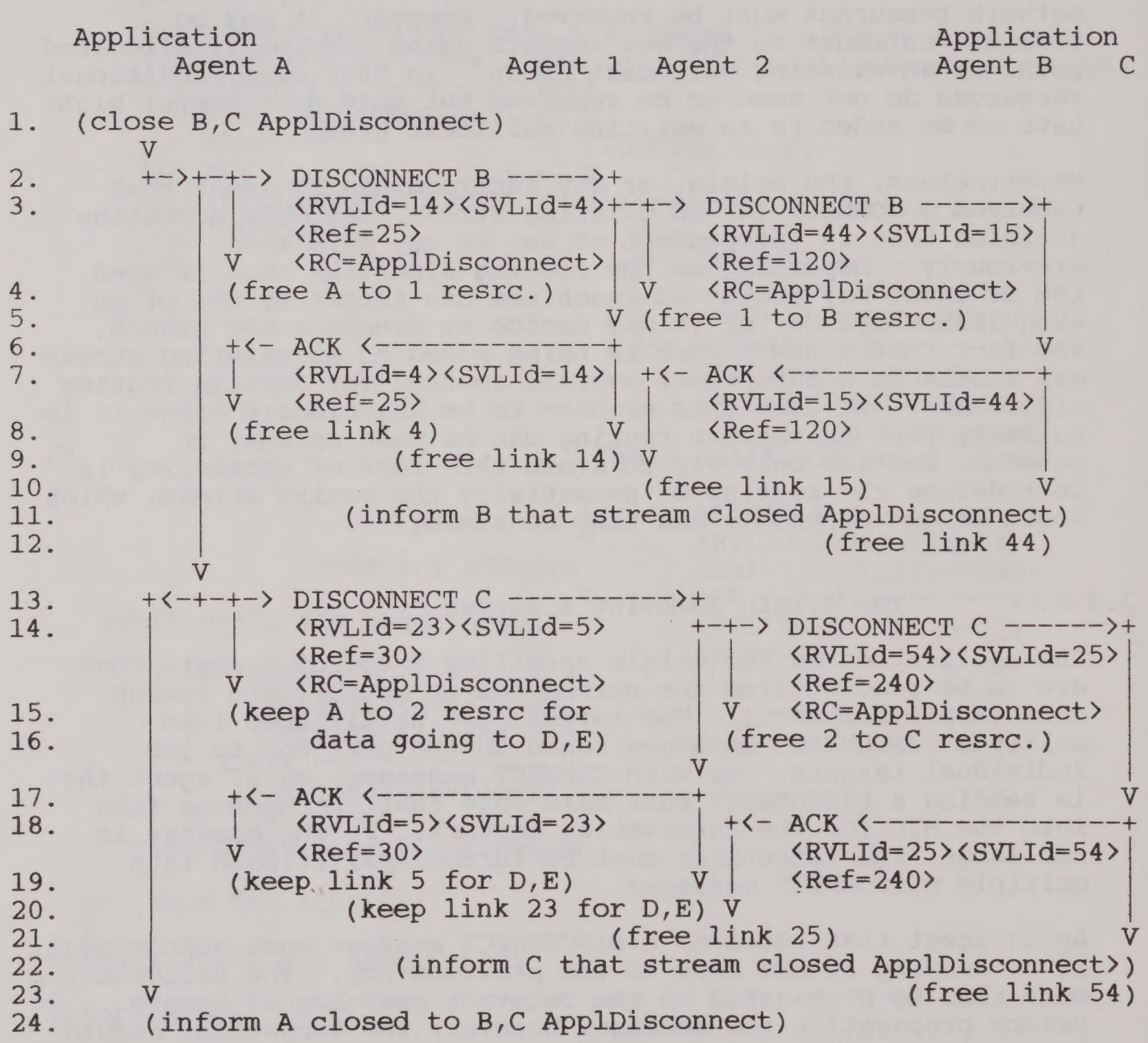


Figure 13. Origin Removing a Target

When the DISCONNECT reaches a target, the target sends an ACK and notifies the application that it is no longer part of the stream and the reason. The application should then inform ST to terminate the stream, and ST should delete the stream from its database after performing any necessary management and accounting functions.

3.3.3. A Target Deleting Itself

The application at the target may inform ST that it wants to be removed from the stream and the appropriate reason code (ApplDisconnect). The agent then forms a REFUSE message with itself as the only entry in the TargetList. The REFUSE is sent back to the origin via the previous-hop. If a stream has multiple targets and one target leaves the stream using this REFUSE mechanism, the stream to the other targets is not affected; the stream continues to exist.

An ST agent that receives such a REFUSE message must acknowledge it by sending an ACK to the next-hop. The target is deleted and, if the next-hop has no remaining targets, then the those resources associated with that next-hop agent may be released. Note that network resources may not actually be released if network multicasting is being used since they may still be required for traffic to other next-hops in the multicast group. The REFUSE must also be propagated back to the previous-hop ST agent.

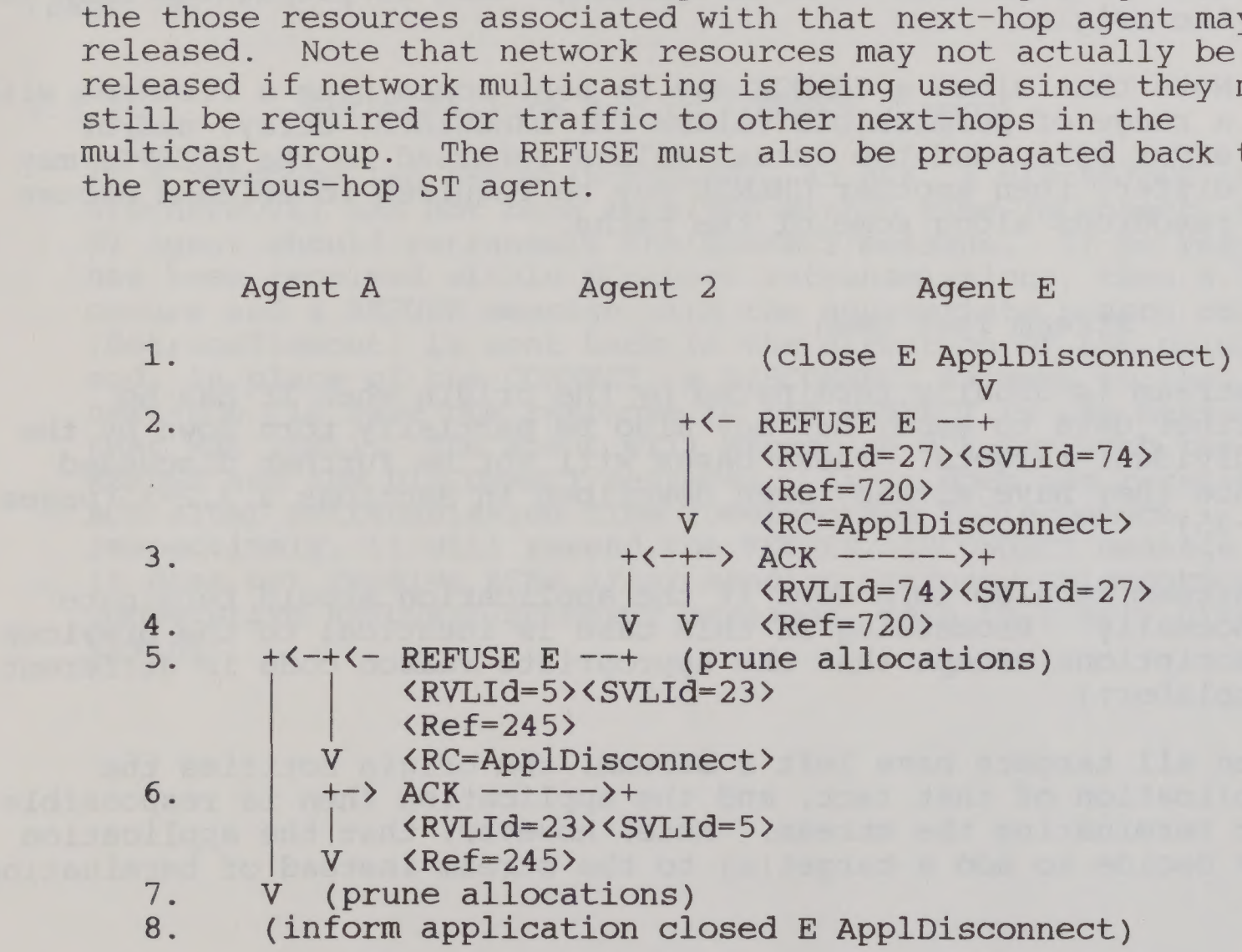


Figure 14. Target Deleting Itself

When the REFUSE reaches the origin, the origin sends an ACK and notifies the application that the target listed in the TargetList is no longer part of the stream. If the stream has no remaining targets, the application may choose to terminate the stream.

3.3.4. Changing the FlowSpec

An application may wish to change the FlowSpec of an established stream. To do so, it informs ST of the new FlowSpec and the list of targets that are to be changed. The origin ST agent then issues one or more CHANGE messages with the new FlowSpec and sends them to the relevant next-hop agents. CHANGE messages are structured and processed similarly to CONNECT messages. A next-hop agent that is an intermediate agent and receives a CHANGE message similarly determines if it can implement the new FlowSpec along the hop to each of its next-hop agents, and if so, it propagates the CHANGE messages along the established paths. If this process succeeds, the CHANGE messages will eventually reach the targets, which will each respond with an ACCEPT message that is propagated back to the origin.

Note that since a CHANGE may be sent containing a FlowSpec with a range of permissible values for bandwidth, delay, and/or error rate, and the actual values returned in the ACCEPTs may differ, then another CHANGE may be required to release excess resources along some of the paths.

3.4. Stream Tear Down

A stream is usually terminated by the origin when it has no further data to send, but may also be partially torn down by the individual targets. These cases will not be further discussed since they have already been described in Sections 3.3.2-3 (pages 33-35).

A stream is also torn down if the application should terminate abnormally. Processing in this case is identical to the previous descriptions except that the appropriate reason code is different (ApplAbort).

When all targets have left a stream, the origin notifies the application of that fact, and the application then is responsible for terminating the stream. Note, however, that the application may decide to add a target(s) to the stream instead of terminating it.

3.5. Exceptional Cases

The previous descriptions covered the simple cases where everything worked. We now discuss what happens when things do not succeed. Included are situations where messages are lost, the requested resources are not available, the routing fails or is inconsistent.

In order for the ST Control Message Protocol to be reliable over an unreliable internetwork, the problems of corruption, duplication, loss, and ordering must be addressed. Corruption is handled through use of checksumming, as described in Section 4 (page 76). Duplication of control messages is detected by assigning a transaction number (Reference) to each control message; duplicates are discarded. Loss is detected using a timeout at the sender; messages that are not acknowledged before the timeout expires are retransmitted; see Section 3.7.6 (page 66). If a message is not acknowledged after a few retransmissions a fault is reported. The protocol does not have significant ordering constraints. However, minor sequencing of control messages for a stream is facilitated by the requirement that the Reference numbers be monotonically increasing; see Section 4.2 (page 78).

3.5.1. Setup Failure due to CONNECT Timeout

If a response (an ERROR-IN-REQUEST, an ACK, a HID-REJECT, or a HID-APPROVE) has not been received within time ToConnect, the ST agent should retransmit the CONNECT message. If no response has been received within NConnect retransmissions, then a fault occurs and a REFUSE message with the appropriate reason code (RetransTimeout) is sent back in the direction of the origin, and, in place of the CONNECT, a DISCONNECT is sent to the next-hop (in case the response to the CONNECT is the message that was lost). The agent will expect an ACK for both the REFUSE and the DISCONNECT messages. If it does not receive an ACK after retransmission time ToRefuse and ToDisconnect respectively, it will resend the REFUSE/DISCONNECT message. If it does not receive ACKs after sending NRefuse/ NDisconnect consecutive REFUSE/DISCONNECT messages, then it simply gives up trying.

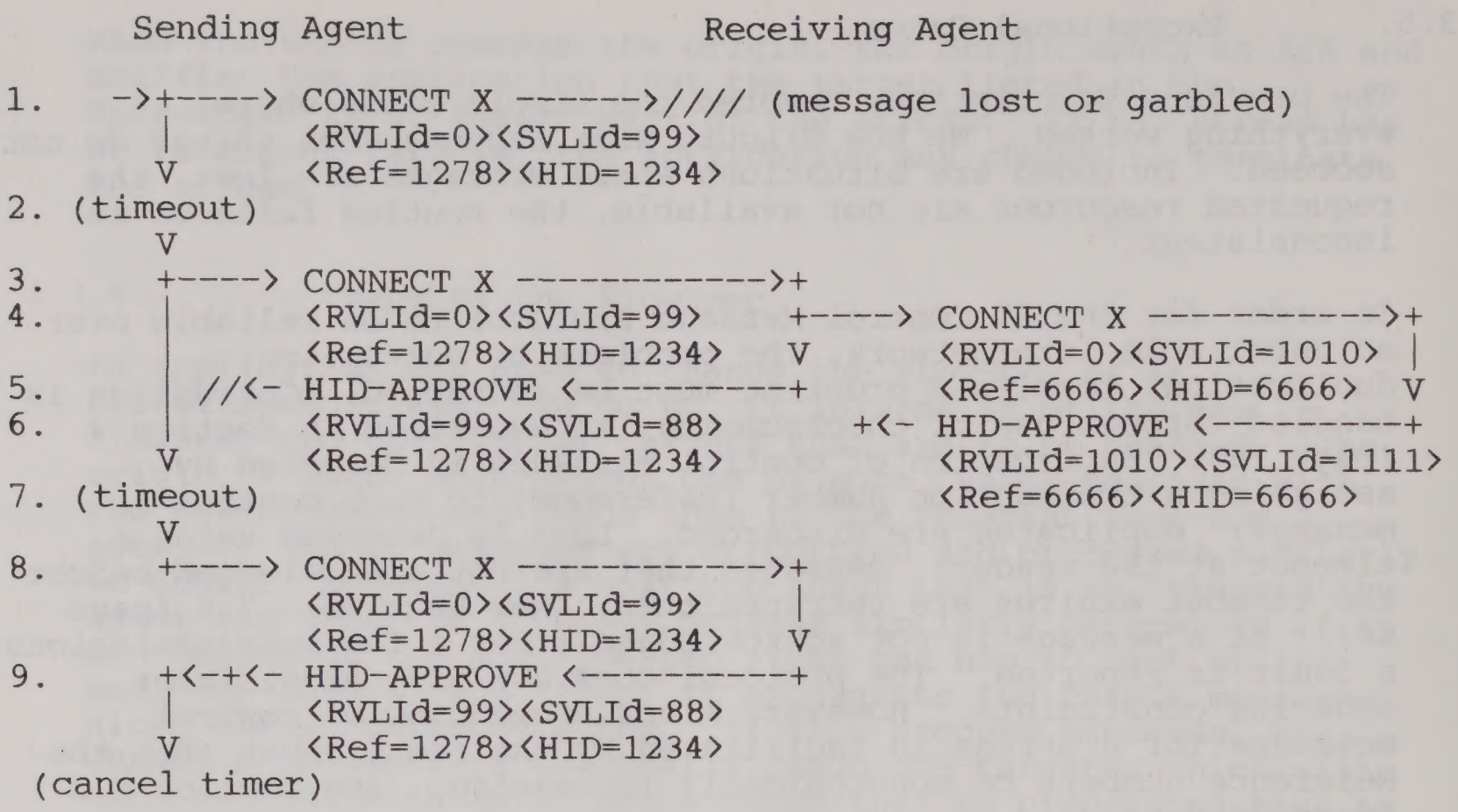


Figure 15. CONNECT Retransmission after a Timeout

3.5.2. Problems due to Routing Inconsistency

When an intermediate agent receives a CONNECT, it selects the next-hop agents based on the TargetList and the networks to which it is connected. If the resulting next-hop to any of the targets is across the same network from which it received the CONNECT (but not the previous-hop itself), there may be a routing problem. However, the routing algorithm at the previous-hop may be optimizing differently than the local algorithm would in the same situation. Since the local ST agent cannot distinguish the two cases, it should permit the setup but send back to the previous-hop agent an informative NOTIFY message with the appropriate reason code (RouteBack), pertinent TargetList, and in the NextHopIPAddress element the address of the next-hop ST agent returned by its routing algorithm.

The agent that receives such a NOTIFY should ACK it. If the agent is using an algorithm that would produce such behavior, no further action is taken; if not, the agent should send a DISCONNECT to the next-hop agent to correct the problem.

Alternatively, if the next-hop returned by the routing function is in fact the previous-hop, a routing inconsistency has been detected. In this case, a REFUSE is sent back to

the previous-hop agent containing an appropriate reason code (RouteInconsist), pertinent TargetList, and in the NextHopIPAddress element the address of the previous-hop. When the previous-hop receives the REFUSE, it will recompute the next-hop for the affected targets. If there is a difference in the routing databases in the two agents, they may exchange CONNECT and REFUSE messages again. Since such routing errors in the internet are assumed to be temporary, the situation should eventually stabilize.

3.5.3. Setup Failure due to a Routing Failure

It is possible for an agent to receive a CONNECT message that contains a known Name, but from an agent other than the previous-hop agent of the stream with that Name. This may be:

1 that two branches of the tree forming the stream have joined back together,

2 a deliberate source routing loop,

3 the result of an attempted recovery of a partially failed stream, or

4 an erroneous routing loop.

The TargetList is used to distinguish the cases 1 and 2 (see also Section 4.2.3.5 (page 107)) by comparing each newly received target with those of the previously existing stream:

- if the IP address of the targets differ, it is case 1;
- if the IP address of the targets match but the source route(s) are different, it is case 2;
- if the target (including any source route) matches a target (including any source route) in the existing stream, it may be case 3 or 4.

It is expected that the joining of branches will become more common as routing decisions are based on policy issues and not just simple connectivity. Unfortunately, there is no good way to merge the two parts of the stream back into a single stream. They must be treated independently with respect to processing in the agent. In particular, a separate state machine is required, the Virtual Link Identifiers and HIDs from the previous-hops and to the next-hops must be different, and duplicate resources must be reserved in both the agent and in any next-hop networks. Processing is the same for a deliberate source routing loop.

The remaining cases requiring recovery, a partially failed stream and an erroneous routing loop, are not easily distinguishable. In attempting recovery of a failed stream, an agent may issue new CONNECT messages to the affected targets; for a full explanation see also Section 3.7.2 (page 51), Failure Recovery. Such a CONNECT may reach an agent downstream of the failure before that agent has received a DISCONNECT from the neighborhood of the failure. Until that agent receives the DISCONNECT, it cannot distinguish between a failure recovery and an erroneous routing loop. That agent must therefore respond to the CONNECT with a REFUSE message with the affected targets specified in the TargetList and an appropriate reason code (StreamExists).

The agent immediately preceding that point, i.e., the latest agent to send the CONNECT message, will receive the REFUSE message. It must release any resources reserved exclusively for traffic to the listed targets. If this agent was not the one attempting the stream recovery, then it cannot distinguish between a failure recovery and an erroneous routing loop. It should repeat the CONNECT after a ToConnect timeout. If after NConnect retransmissions it continues to receive REFUSE messages, it should propagate the REFUSE message toward the origin, with the TargetList that specifies the affected targets, but with a different error code (RouteLoop).

The REFUSE message with this error code (RouteLoop) is propagated by each ST agent without retransmitting any CONNECT messages. At each agent, it causes any resources reserved exclusively for the listed targets to be released. The REFUSE will be propagated to the origin in the case of an erroneous routing loop. In the case of stream recovery, it will be propagated to the ST agent that is attempting the recovery, which may be an intermediate agent or the origin itself. In the case of a stream recovery, the agent attempting the recovery may issue new CONNECT messages to the same or to different next-hops.

If an agent receives both a REFUSE message and a DISCONNECT message with a target in common then it can release the relevant resources and propagate neither the REFUSE nor the DISCONNECT (however, we feel that it is unlikely that most implementations will be able to detect this situation).

If the origin receives such a REFUSE message, it should attempt to send a new CONNECT to all the affected targets. Since routing errors in an internet are assumed to be temporary, the new CONNECTs will eventually find acceptable routes to the targets, if one exists. If no further routes exist after NRetryRoute tries, the application should be

informed so that it may take whatever action it deems necessary.

3.5.4. Problems in Reserving Resources

If the network or ST agent resources are not available, an ST agent may preempt one or more streams that have lower precedence than the one being created. When it breaks a lower precedence stream, it must issue REFUSE and DISCONNECT messages as described in Sections 4.2.3.15 (page 122) and 4.2.3.6 (page 110). If there are no streams of lower precedence, or if preempting them would not provide sufficient resources, then the stream cannot be accepted by the ST agent.

If an intermediate agent detects that it cannot allocate the necessary resources, then it sends a REFUSE that contains an appropriate reason code (CantGetResrc) and the pertinent TargetList to the previous-hop ST agent. For further study are issues of reporting what resources are available, whether the resource shortage is permanent or transitory, and in the latter case, an estimate of how long before the requested resources might be available.

3.5.5. Setup Failure due to ACCEPT Timeout

An ST agent that propagates an ACCEPT message backward toward the origin expects an ACK from the previous-hop. If it does not receive an ACK within a timeout, called ToAccept, it will retransmit the ACCEPT. If it does not receive an ACK after sending a number, called NAccept, of ACCEPT messages, then it will replace the ACCEPT with a REFUSE, and will send a DISCONNECT in the direction toward the target. Both the REFUSE and DISCONNECT will identify the affected target(s) and specify an appropriate reason code (AcceptTimeout). Both are also retransmitted until ACKed with timeout ToRefuse/ ToDisconnect and retransmit count NRefuse/NDisconnect. If they are not ACKed, the agent simply gives up, letting the failure detection mechanism described in Section 3.7.1 (page 48) take care of any cleanup.

3.5.6. Problems Caused by CHANGE Messages

An application must exercise care when changing a FlowSpec to prevent a failure. A CHANGE might fail for two reasons. The request may be for a larger amount of network resources when those resources are not available; this failure may be prevented by requiring that the current level of service be contained within the ranges of the FlowSpec in the CHANGE.

Alternatively, the local network might require all the former resources to be released before the new ones are requested and, due to unlucky timing, an unrelated request for network resources might be processed between the time the resources are released and the time the new resources are requested, so that the former resources are no longer available. There is not much that an application or ST can do to prevent such failures.

If the attempt to change the FlowSpec fails then the ST agent where the failure occurs must intentionally break the stream and invoke the stream recovery mechanism using REFUSE and DISCONNECT messages; see Section 3.7.2 (page 51). Note that the reserved resources after the failure of a CHANGE may not be the same as before, i.e., the CHANGE may have been partially completed. The application is responsible for any cleanup (another CHANGE).

3.5.7. Notification of Changes Forced by Failures

NOTIFY is issued by a an ST Agent to inform upsteam agents and the origin that resource allocation changes have occurred after a stream was established. These changes occur when network components fail and when competing streams preempt resources previously reserved by a lower precedence stream. We also anticipate that NOTIFY can be used in the future when additional resources become available, as is the case when network components recover or when higher precedence streams are deleted.

NOTIFY is also used to inform upstream agents that a routing anomaly has occurred. Such an example was cited in Section 3.5.2 (page 38), where an agent notices that the next-hop agent is on the same network as the previous-hop agent; the anomaly is that the previous-hop should have connected directly to the next-hop without using an intermediate agent. Delays in propagating host status and routing information can cause such anomalies to occur. NOTIFY allows ST to correct automatically such mistakes.

NOTIFY reports a FlowSpec that reflects that revised guarantee that can be promised to the stream. NOTIFY also

identifies those targets affected by the change. In this way, NOTIFY is similar to ACCEPT. NOTIFY includes a ReasonCode to identify the event that triggered the notification. It also includes a TargetList, rather than a single Target, since a single event can affect a branch leading to several targets.

NOTIFY is relayed by the ST agents back toward the origin, along the path established by the CONNECT but in the reverse direction. NOTIFY must be acknowledged with an ACK at each hop. If intermediate agent corrects the situation without causing any disruption to the data flow or guarantees, it can choose to drop the notification message before it reaches the origin. If the originating agent receives a NOTIFY, it is then expected to adjust its own processing and data rates, and to submit any required CHANGE requests. As with ACCEPT, the FlowSpec is not modified on this trip from the target back to the origin. It is up to the origin to decide whether a CHANGE should be submitted. (However, even though the FlowSpec has not been modified, the situation reported in the

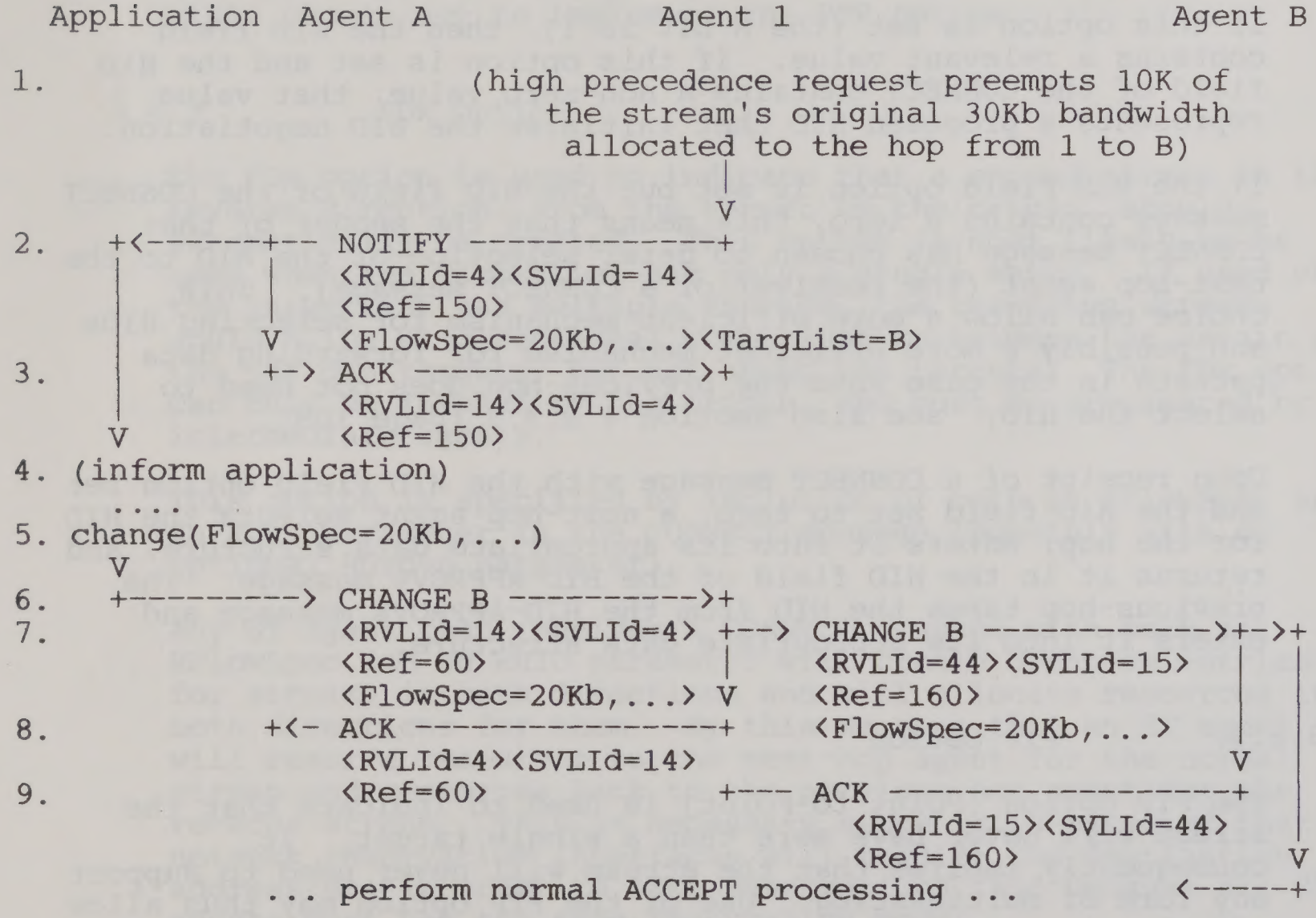


Figure 16. Processing NOTIFY Messages

notify may have prevented the ST agents from meeting the original guarantees.)

3.6. Options

Several options are defined in the CONNECT message. The special processing required to support each will be described in the following sections. The options are independent, i.e., can be set to one (1, TRUE) or zero (0, FALSE) in any combination. However, the effect and implementation of the options is NOT necessarily independent, and not all combinations are supported.

3.6.1. HID Field Option

The sender of a CONNECT message may or not specify an HID in the HID field. If the HID Field option of the CONNECT message is not set (the H bit is 0), then the HID field does not contain relevant information and should be ignored.

If this option is set (the H bit is 1), then the HID field contains a relevant value. If this option is set and the HID field of the CONNECT contains a non-zero value, that value represents a proposed HID that initiates the HID negotiation.

If the HID Field option is set but the HID field of the CONNECT message contains a zero, this means that the sender of that CONNECT message has chosen to defer selection of the HID to the next-hop agent (the receiver of a CONNECT message). This choice can allow a more efficient mechanism for selecting HIDs and possibly a more efficient mechanism for forwarding data packets in the case when the previous-hop does not need to select the HID; see also Section 4.2.3.5 (page 105).

Upon receipt of a CONNECT message with the HID Field option set and the HID field set to zero, a next-hop agent selects the HID for the hop, enters it into its appropriate data structure, and returns it in the HID field of the HID-APPROVE message. The previous-hop takes the HID from the HID-APPROVE message and enters it into its appropriate data structure.

3.6.2. PTP Option

The PTP option (Point-to-Point) is used to indicate that the stream will never have more than a single target. It consequently implies that the stream will never need to support any form of multicasting. Use of the PTP option may thus allow efficiencies in the way the stream is built or is

managed. Specifically, the ST agents do not need to request that the intervening networks allocate multicast groups to support this stream.

The PTP option can only be set to one (1) by the origin, and must be the same for the entire stream (i.e., propagated by ST agents). The details of what this option does are implementation specific, and do not affect the protocol very much.

If the application attempts to add a new target to an existing stream that was created with the PTP option set to one (1), the application should be informed of the error with an ERROR-IN-REQUEST message with the appropriate reason code. If a CONNECT is received whose TargetList contains more than a single entry, an ERROR-IN-REQUEST message with the appropriate reason code (PTPError) should be returned to the previous-hop agent (note that such a CONNECT should never be received if the origin both implements the PTP option and is functioning properly).

As implied in the last paragraph, a subsetted implementation might choose not to implement the PTP option.

3.6.3. FDx Option

The FDx option is used to indicate that a second stream in the reverse direction, from the target to the origin, should automatically be created. This option is most likely to be used when the TargetList has only a single entry. If used when the TargetList has multiple entries, the resulting streams would allow bi-directional communication between the origin and the various targets, but not among the targets. The FDx option can only be invoked by the origin, and must be propagated by intermediate agents.

This option is specified by inclusion of both an RFlowSpec and an RHID parameter in the CONNECT message (possibly with an optional RGroup parameter).

Any ST agent that receives a CONNECT message with both an RFlowSpec and an RHID parameter will create database entries for streams in both directions and will allocate resources in both directions for them. By this we mean that an ST agent will reserve resources to the next-hop agent for the normal stream and resources back to the previous-hop agent for the reverse stream. This is necessary since it is expected that network reservation interfaces will require the destination address(es) in order to make reservations, and because all ST agents must use the same reservation model.

The target agent will select a Name for the reverse stream and return it (in the RName parameter) and the resulting FlowSpec (in the RFlowSpec parameter) of the ACCEPT message. Each agent that processes the ACCEPT will update its partial stream database entry for the reverse stream with the Name contained in the RName parameter. We assume that the next higher protocol layer will use the same SAP for both streams.

3.6.4. NoRecovery Option

The NoRecovery option is used to indicate that ST agents should not attempt recovery in case of network or component failure. If a failure occurs, the origin will be notified via a REFUSE message and the target(s) via a DISCONNECT, with an appropriate reason code of "failure" (i.e., one of DropFailAgt, DropFailHst, DropFailIfc, DropFailNet, IntfcFailure, NetworkFailure, STAgentFailure, FailureRecovery). They can then decide whether to wait for the failed component to be fixed, or drop the target via DISCONNECT/REFUSE messages. The NoRecovery option can only be set to one (1) by the origin, and must be the same for the entire stream.

3.6.5. RevChrg Option

The RevChrg option bit in the FlowSpec is set to one (1) by the origin to request that the target(s) pay any charges associated with the stream (to the target(s)); see Section 4.2.2.3 (page 83). If the target is not willing to accept charges, the bit should be set to zero (0) by the target before returning the FlowSpec to the origin in an ACCEPT message.

If the FDx option is also specified, the target pays charges for both streams.

3.6.6. Source Route Option

The Source Route Option may be used both for diagnostic purposes, and, in those hopefully infrequent cases where the standard routing mechanisms do not produce paths that satisfy some policy constraint, to allow the origin to prespecify the ST agents along the path to the target(s). The idea is that the origin can explicitly specify the path to a target, either strictly hop-by-hop or more loosely by specification of one or more agents through which the path must pass.

The option is specified by including source routing information in the Target structure. A target may contain zero or more SrcRoute options; when multiple options are present, they are processed in the order in which they occur. The parameter code indicates whether the portion of the path contained in the parameter is of the strict or loose variety.

Since portions of a path may pass through portions of an internet that does not support ST agents, there are also forms of the SrcRoute option that are converted into the

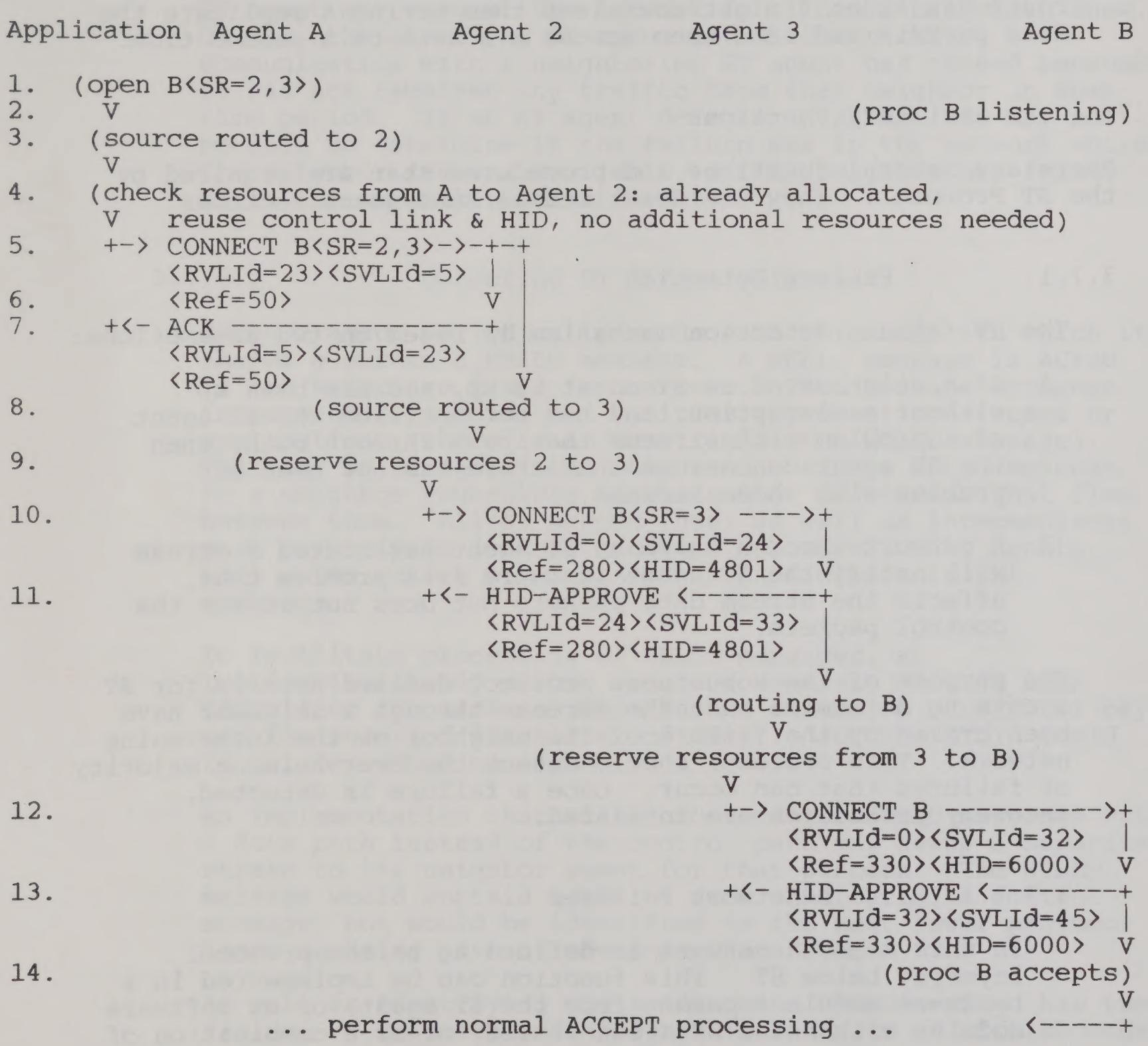


Figure 17. Source Routing Option

corresponding IP Source Routing options by the ST agent that performs the encapsulation.

The SrcRoute option is usually selected by the origin, but may be used by intermediate agents if specified as a result of the routing function.

For example, in the topology of Figure 2, if A wants to add B back into the stream, its routing function might decide that the best path is via Agent 3. Since the data is already being multicast across the network connected to C, D, and E, the route via Agent 3 might cost less than having A replicate the data packets and send them across A's network a second time.

3.7. Ancillary Functions

There are several functions and procedures that are required by the ST Protocol. They are described in subsequent sections.

3.7.1. Failure Detection

The ST failure detection mechanism is based on two assumptions:

1 If a neighbor of an ST agent is up, and has been up without a disruption, and has not notified the ST agent of a problem with streams that pass through both, then the ST agent can assume that there has not been any problem with those streams.

2 A network through which an ST agent has routed a stream will notify the ST agent if there is a problem that affects the stream data packets but does not affect the control packets.

The purpose of the robustness protocol defined here is for ST agents to determine that the streams through a neighbor have been broken by the failure of the neighbor or the intervening network. This protocol should detect the overwhelming majority of failures that can occur. Once a failure is detected, recovery procedures are initiated.

3.7.1.1. Network Failures

In this memo, a network is defined to be the protocol layer(s) below ST. This function can be implemented in a hardware module separate from the ST agent, or as software modules within the ST agent itself, or as a combination of

both. This specification and the robustness protocol do not differentiate between these alternatives.

An ST agent can detect network failures by two mechanisms; the network can report a failure, or the ST agent can discover a failure by itself. They differ in the amount of information that ST agent has available to it in order to make a recovery decision. For example, a network may be able to report that reserved bandwidth has been lost and the reason for the loss and may also report that connectivity to the neighboring ST agent remains intact. In this case, the ST agent may request the network to allocate bandwidth anew. On the other hand, an ST agent may discover that communication with a neighboring ST agent has ceased because it has not received any traffic from that neighbor in some time period. If an ST agent detects a failure, it may not be able to determine if the failure was in the network while the neighbor remains available, or the neighbor has failed while the network remains intact.

3.7.1.2. Detecting ST Stream Failures

Each ST agent periodically sends each neighbor with which it shares a stream a HELLO message. A HELLO message is ACKed if the Reference field is non-zero. This message exchange is between ST agents, not entities representing streams or applications (there is no Name field in a HELLO message). That is, an ST agent need only send a single HELLO message to a neighbor regardless of the number of streams that flow between them. All ST agents (host as well as intermediate) must participate in this exchange. However, only agents that share active streams need to participate in this exchange.

To facilitate processing of HELLO messages, an implementation may either create a separate Virtual Link Identifier for each neighbor having an active stream, or may use the reserved identifier of one (1) for the SVLId field in all its HELLO messages.

An implementation that wishes to send its HELLO messages via a data path instead of the control path may setup a separate stream to its neighbor agent for that purpose. The HELLO message would contain a HID of zero, indicating a control message, but would be identified to the next lower protocol layer as being part of the separate stream.

As well as identifying the sender, the HELLO message has two fields; a HelloTimer field that is in units of milliseconds modulo the maximum for the field size, and a

Restarted bit specifying that the ST agent has been restarted recently. The HelloTimer must appear to be incremented every millisecond whether a HELLO message is sent or not, but it is allowable for an ST agent to create a new HelloTimer only when it sends a HELLO message. The HelloTimer wraps around to zero after reaching the maximum value. Whenever an ST agent suffers a catastrophic event that may result in it losing ST state information, it must reset its HelloTimer to zero and must set the Restarted bit for the following HelloTimerHoldDown seconds.

An ST agent must send HELLO messages to its neighbor with a period shorter than the smallest RecoveryTimeout parameter of the FlowSpecs of all the active streams that pass between the two agents, regardless of direction. This period must be smaller by a factor, called HelloLossFactor, which is at least as large as the greatest number of consecutive HELLO messages that could credibly be lost while the communication between the two ST agents is still viable.

An ST agent may send simultaneous HELLO messages to all its neighbors at the rate necessary to support the smallest RecoveryTimeout of any active stream. Alternately, it may send HELLO messages to different neighbors independently at different rates corresponding to RecoveryTimeouts of individual streams.

The agent that receives a HELLO message expects to receive at least one new HELLO message from a neighbor during the RecoveryTimeout of every active stream through that neighbor. It can detect duplicate or delayed HELLO messages by saving the HelloTimer field of the most recent valid HELLO message from that neighbor and comparing it with the HelloTimer field of incoming HELLO messages. It will only accept an incoming HELLO message from that neighbor if it has a HelloTimer field that is greater than the most recent valid HELLO message by the time elapsed since that message was received plus twice the maximum likely delay variance from that neighbor. If the ST agent does not receive a valid HELLO message within the RecoveryTimeout of a stream, it must assume that the neighboring ST agent or the communication link between the two has failed and it must initiate stream recovery activity.

Furthermore, if an ST agent receives a HELLO message that contains the Restarted bit set, it must assume that the sending ST agent has lost its ST state. If it shares streams with that neighbor, it must initiate stream recovery activity. If it does not share streams with that neighbor, it should not attempt to create one until that

bit is no longer set. If an ST agent receives a CONNECT message from a neighbor whose Restarted bit is still set, it must respond with ERROR-IN-REQUEST with the appropriate reason code (RemoteRestart). If it receives a CONNECT message while its own Restarted bit is set, it must respond with ERROR-IN-REQUEST with the appropriate reason code (RestartLocal).

3.7.1.3. Subset

This failure detection mechanism subsets by reducing the complexity of the timing and decisions. A subsetted ST agent sends HELLO messages to all its ST neighbors regardless of whether there is an active ST stream between them or not. The RecoveryTimeout parameter of the FlowSpec is ignored and is assumed to be the DefaultRecoveryTimeout. Note that this implies that a REFUSE should be sent for all CONNECT or CHANGE messages whose RecoveryTimeout is less than DefaultRecoveryTimeout. An ST agent will accept an incoming HELLO message if it has a HelloTimer field that is greater than the most recent valid HELLO message by DefaultHelloFactor times the time elapsed since that message was received.

3.7.2. Failure Recovery

Streams can fail from various causes; an ST agent can break, a network can break, or an ST agent can intentionally break a stream in order to give the stream's resources to a higher precedence stream. We can envision several approaches to recovery of broken streams, and we consider the one described here the simplest and therefore the most likely to be implemented and work.

If an intermediate agent fails or a network or part of a network fails, the previous-hop agent and the various next-hop agents will discover the fact by the failure detection mechanism described in Section 3.7.1 (page 48). An ST agent that intentionally breaks a stream obviously knows of the event.

The recovery of an ST stream is a relatively complex and time consuming effort because it is designed in a general manner to operate across a large number of networks with diverse characteristics. Therefore, it may require information to be distributed widely, and may require relatively long timers. On the other hand, since a network is a homogeneous system, failure recovery in the network may be a relatively faster and simpler operation. Therefore an ST agent that detects a failure should attempt to fix the network failure before

attempting recovery of the ST stream. If the stream that existed between two ST agents before the failure cannot be reconstructed by network recovery mechanisms alone, then the ST stream recovery mechanism must be invoked.

If stream recovery is necessary, the different ST agents may need to perform different functions, depending on their relation to the failure.

An intermediate agent that breaks the stream intentionally sends DISCONNECT messages with the appropriate reason code (StreamPreempted) toward the affected targets. If the NoRecovery option is selected, it sends a REFUSE message with the appropriate reason code(StreamPreempted) toward the origin. If the NoRecovery option is not selected, then this agent attempts recovery of the stream, as described below.

A host agent that is a target of the broken stream or is itself the next-hop of the failed component should release resources that are allocated to the stream, but should maintain the internal state information describing the stream. It should inform any next higher protocol of the failure. It is appropriate for that protocol to expect that the stream will be fixed shortly by some alternate path and so maintain, for some time period, whatever information in the ST layer, the next higher layer, and the application is necessary to reactivate quickly entries for the stream as the alternate path develops. The agent should use a timeout to delete all the stream information in case the stream cannot be fixed in a reasonable time.

An intermediate agent that is a next-hop of a failure that was not due to a preemption should first verify that there was a failure. It can do this using STATUS messages to query its upstream neighbor. If it cannot communicate with that neighbor, then it should first send a REFUSE message with the appropriate reason code of "failure" to the neighbor to speed up the failure recovery in case the hop is unidirectional, i.e., the neighbor can hear the agent but the agent cannot hear the neighbor. The ST agent detecting the failure must then send DISCONNECT messages with the same reason code toward the targets. The intermediate agents process this DISCONNECT message just like the DISCONNECT that tears down the stream. However, a target ST agent that receives a DISCONNECT message with the appropriate reason code (StreamPreempted, or "failure") will maintain the stream state and notify the next higher protocol of the failure. In effect, these DISCONNECT messages tear down the stream from the point of the failure to the targets, but inform the targets that the stream may be fixed shortly.

An ST agent that is the previous-hop before the failed component first verifies that there was a failure by querying the downstream neighbor using STATUS messages. If the neighbor has lost its state but is available, then the ST agent may reconstruct the stream if the NoRecovery option is not selected, as described below. If it cannot communicate with the next-hop, then the agent detecting the failure releases any resources that are dedicated exclusively to sending data on the broken branch and sends a DISCONNECT message with the appropriate reason code ("failure") toward the affected targets. It does so to speed up failure recovery in case the communication may be unidirectional and this message might be delivered successfully.

If the NoRecovery option is selected, then the ST agent that detects the failure sends a REFUSE message with the appropriate reason code ("failure") to the previous-hop. If it is breaking the stream intentionally, it sends a REFUSE message with the appropriate reason code (StreamPreempted) to the previous-hop. The TargetList in these messages contains all the targets that were reached through the broken branch. Multiple REFUSE messages may be required if the PDU is too long for the MTU of the intervening network. The REFUSE message is propagated all the way to the origin, which can attempt recovery of the stream by sending a new CONNECT to the affected targets. The new CONNECT will be treated by intermediate ST agents as an addition of new targets into the established stream.

If the NoRecovery option is not selected, the ST agent that breaks the stream intentionally or is the previous-hop before the failed component can attempt recovery of the stream. It does so by issuing a new CONNECT message to the affected targets. If the ST agent cannot find new routes to some targets, or if the only route to some targets is through the previous-hop, then it sends one or more REFUSE messages to the previous-hop with the appropriate reason code ("failure" or StreamPreempted) specifying the affected targets in the TargetList. The previous-hop can then attempt recovery of the stream by issuing a CONNECT to those targets. If it cannot find an appropriate route, it will propagate the REFUSE message toward the origin.

Regardless of which agent attempts recovery of a damaged stream, it will issue one or more CONNECT messages to the affected targets. These CONNECT messages are treated by intermediate ST agents as additions of new targets into the established stream. The FlowSpecs of the new CONNECT messages should be the same as the ones contained in the most recent CONNECT or CHANGE messages that the ST agent had sent toward the affected targets when the stream was operational.

The reconstruction of a broken stream may not proceed smoothly. Since there may be some delay while the information concerning the failure is propagated throughout an internet, routing errors may occur for some time after a failure. As a result, the ST agent attempting the recovery may receive REFUSE or ERROR-IN-REQUEST messages for the new CONNECTs that are caused by internet routing errors. The ST agent attempting the recovery should be prepared to resend CONNECTs before it succeeds in reconstructing the stream. If the failure partitions the internet and a new set of routes cannot be found to the targets, the REFUSE messages will eventually be propagated to the origin, which can then inform the application so it can decide whether to terminate or to continue to attempt recovery of the stream.

The new CONNECT may at some point reach an ST agent downstream of the failure before the DISCONNECT does. In this case, the agent that receives the CONNECT is not yet aware that the stream has suffered a failure, and will interpret the new CONNECT as resulting from a routing failure. It will respond with an ERROR-IN-REQUEST message with the appropriate reason code (StreamExists). Since the timeout that the ST agents immediately preceding the failure and immediately following the failure are approximately the same, it is very likely that the remnants of the broken stream will soon be torn down by a DISCONNECT message with the appropriate reason code ("failure"). Therefore, the ST agent that receives the ERROR-IN-REQUEST message with reason code (StreamExists) should retransmit the CONNECT message after the ToConnect timeout expires. If this fails again, the request will be retried for NConnect times. Only if it still fails will the ST agent send a REFUSE message with the appropriate reason code (RouteLoop) to its previous-hop. This message will be propagated back to the ST agent that is attempting recovery of the damaged stream. That ST agent can issue a new CONNECT message if it so chooses. The REFUSE is matched to a CONNECT message created by a recovery operation through the LnkReference field in the CONNECT.

ST agents that have propagated a CONNECT message and have received a REFUSE message should maintain this information for some period of time. If an agent receives a second CONNECT message for a target that recently resulted in a REFUSE, that agent may respond with a REFUSE immediately rather than attempting to propagate the CONNECT. This has the effect of pruning the tree that is formed by the propagation of CONNECT messages to a target that is not reachable by the routes that are selected first. The tree will pass through any given ST agent only once, and the stream setup phase will be completed faster.

The time period for which the failure information is maintained must be consistent with the expected lifetime of that information. Failures due to lack of reachability will remain relevant for time periods large enough to allow for network reconfigurations or repairs. Failures due to routing loops will be valid only until the relevant routing information has propagated, which can be a short time period. Lack of bandwidth resulting from over-allocation will remain valid until streams are terminated, which is an unpredictable time, so the time that such information is maintained should also be short.

If a CONNECT message reaches a target, the target should as efficiently as possible use the state that it has saved from before the stream failed during recovery of the stream. It will then issue an ACCEPT message toward the origin. The ACCEPT message will be intercepted by the ST agent that is attempting recovery of the damaged stream, if not the origin. If the FlowSpec contained in the ACCEPT specifies the same selection of parameters as were in effect before the failure, then the ST agent that is attempting recovery will not propagate the ACCEPT. If the selections of the parameters are different, then the agent that is attempting recovery will send the origin a NOTIFY message with the appropriate reason code (FailureRecovery) that contains a FlowSpec that specifies the new parameter values. The origin may then have to change its data generation characteristics and the stream's parameters with a CHANGE message to use the newly recovered subtree.

3.7.2.1. Subset

Subsets of this mechanism may reduce the functionality in the following ways. A host agent might not retain state describing a stream that fails with a DISCONNECT message with the appropriate reason code ("failure" or StreamPreempted).

An agent might force the NoRecovery option always to be set. In this case, it will allow the option to be propagated in the CONNECT message, but will propagate the REFUSE message with the appropriate reason code ("failure" or StreamPreempted) without attempting recovery of the damaged stream.

If an ST agent allows stream recovery and attempts recovery of a stream, it might choose a FlowSpec to specify exactly the current values of the parameters, with no ranges or options.

3.7.3. A Group of Streams

There may be a need to associate related streams. The Group mechanism is simply an association technique that allows ST agents to identify the different streams that are to be associated. Streams are in the same Group if they have the same Group Name in the GroupName field of the (R)Group parameter. At this time there are no ST control messages that modify Groups. Group Names have the same format as stream Names, and can share the same name space. A stream that is a member of a Group can specify one or more (Subgroup Identifier, Relation) tuples. The Relation specifies how the members of the Subgroup of the Group are related. The Subgroups Identifiers need only be unique within the Group.

Streams can be associated into Groups to support activities that deal with a number of streams simultaneously. The operation of Groups of streams is a matter for further study, and this mechanism is provided to support that study. This mechanism allows streams to be identified as belonging to a given Group and Subgroup, but in order to have any effect, the behavior that is expected of the Relation must be implemented in the ST agents. Possible applications for this mechanism include the following:

- Associating streams that are part of a floor-controlled conference. In this case, only one origin can send data through its stream at any given time. Therefore, at any point where more than one stream passes through a branch or network, only enough bandwidth for one stream needs to be allocated.

- Associating streams that cannot exist independently. An example of this may be the various streams that carry the audio, video, and data components of a conference, or the various streams that carry data from the different participants in a conference. In this case, if some ST agent must preempt more than a single stream, and it has selected any one of the streams so associated, then it should also preempt the rest of the members of that Subgroup rather than preempting any other streams.

- Associating streams that must not be completed independently. This example is similar to the preceding one, but relates to the stream setup phase. In this example, any single member of a Subgroup of streams need not be completed unless the rest are also completed. Therefore, if one stream becomes blocked, all the others will also be blocked. In this case, if there are not enough resources to support all the conferences that are attempted, some number of the conferences will complete

and other will be blocked, rather than all conferences be partially completed and partially blocked.

This document assumes that the creation and membership of the Group will be managed by the next protocol above ST, with the assistance of ST. For example, the next higher protocol would request ST to create a unique Group Name and a set of Subgroups with specified characteristics. The next higher protocol would distribute this information to the other participants that were to be members of the Group. Each would transfer the Group Name, Subgroups, and Relations to the ST layer, which would simply include them in the stream state.

3.7.3.1. Group Name Generator

This facility is provided so that an application or higher layer protocol can obtain a unique Group Name from the ST layer. This is a mechanism for the application to request the allocation of a Group Name that is independent of the request to create a stream. The Group Name is used by the application or higher layer protocol when creating the streams that are to be part of a group. All that is required is a function of the form:

```
AllocateGroupName()
   -> result, GroupName
```

A corresponding function to release a Group Name is also desirable; its form is:

```
ReleaseGroupName( GroupName )
   -> result
```

3.7.3.2. Subset

Since Groups are currently intended to support experimentation, and it is not clear how best to use them, it is appropriate for an implementation not to support Groups. At this time, a subsetted ST agent may ignore the Group parameter. It is expected that in the future, when Groups transition from being an experimental concept to an operational one, it may be the case that such subsetting will no longer be acceptable. At that time, a new subsetting option may be defined.

3.7.4. HID Negotiation

Each data packet must carry a value to identify the stream to which it belongs, so that forwarding can be performed. Conceptually, this value could be the Name of the stream. A shorthand identifier is desirable for two reasons. First, since each data packet must carry this identifier, network bandwidth efficiency suggests that it be as small as possible. This is particularly important for applications that use small data packets, and that use low bandwidth networks, such as voice across packet radio networks. Second, the operation of mapping this identifier into a data object that contains the forwarding information must be performed at each intermediate ST agent in the stream. To minimize delay and processing overhead, this operation should be as efficient as possible. Most likely, this identifier will be used to index into an internal table. To meet these goals, ST has chosen to use a 16-bit hop-by-hop identifier (HID). It is large enough to handle the foreseen number of streams during the expected life of the protocol while small enough not to preclude its use as a forwarding table index. Note, however, that HID 0 is reserved for control messages, and that HIDs 1-3 are also reserved for future use.

When ST makes use of multicast ability in networks that provide it, a data packet multicast by an ST agent will be received identically by several next-hop ST agents. In a multicast environment, the HID must be selected either by some network-wide mechanism that selects unique identifiers, or it must be selected by the sender of the CONNECT message. Since we feel any network-wide mechanism is outside the scope of this protocol, we propose that the previous-hop agent select the HID and send it in the CONNECT message (with the HID Field option set, see Section 3.6.1 (page 44)) subject to the approval of the next-hop agents. We call this "HID negotiation".

As an origin ST agent is creating a stream or as an intermediate agent is propagating a CONNECT message, it must make a routing decision to determine which targets will be reached through which next-hop ST agents. In some cases, several next-hops can be reached through a network that supports multicast delivery. If so, those next-hops will be made members of a multicast group and data packets will be sent to the group. Different CONNECT messages are sent to the several next-hops even if the data packets will be sent to the multicast group, because the CONNECT messages contain different TargetLists and are acknowledged and accepted separately. However, the HID contained by the different CONNECT message must be identical. The ST agent selects a 16-bit quantity to be the HID and inserts it into each

CONNECT message that is then sent to the appropriate next-hop.

The next-hop agents that receive the CONNECT messages must propagate the CONNECT messages toward the targets, but must also look at the HID and decide whether they can approve it. An ST agent can only receive data packets with a given HID if they belong to a single stream. If the ST agent already has an established stream that uses the proposed HID, this is a HID collision, and the agent cannot approve the HID for the new stream. Otherwise the agent can approve the HID. If it can approve the HID, then it must make note of that HID and it must respond with a HID-APPROVE message (unless it can immediately respond with an ERROR-IN-REQUEST or a REFUSE). If it cannot approve the HID then it must respond with a HID-REJECT message.

An agent that sends a CONNECT message with the H bit set awaits its acknowledgment message (which could be a HID-ACCEPT, HID-REJECT, or an ERROR-IN-REQUEST) from the next-hops independently of receiving ACCEPT messages. If it does not receive an acknowledgment within timeout ToConnect, it will resend the CONNECT. If each next-hop agent responds with a HID-ACCEPT, this implies that they have each approved of the HID, so it can be used for all subsequent data packets. If one or more next-hops respond with an HID-REJECT, then the agent that selected the HID must select another HID and send it to each next-hop in a set of HID-CHANGE messages. The next-hop agents must respond to (and thus acknowledge) these HID-CHANGE messages with either a HID-ACCEPT or a HID-REJECT (or, in the case of an error, an ERROR-IN-REQUEST, or a REFUSE if the next-hop agent wants to abort the HID negotiation process after rejecting NHIDAbort proposed HIDs). If the agent does not receive such a response within timeout ToHIDChange, it will resend the HID-CHANGE up to NHIDChange times. If any next-hop agents respond with a REFUSE message that specifies all the targets that were included in the corresponding CONNECT, then that next-hop is removed from the negotiation. The overall negotiation is complete only when the agent receives a HID-ACCEPT to the same proposed HID from all the next-hops that do not respond with an ERROR-IN-REQUEST or a REFUSE.

This negotiation may continue an indeterminate length of time. In fact, the CONNECT messages could propagate to the targets and their ACCEPT messages may potentially propagate back to the origin before the negotiation is complete. If this were permitted, the origin would not be aware of the incomplete negotiation and could begin to send data packets. Then the agent that is attempting to select a HID would have to discard any data rather than sending it to the next-hops since it might not have a valid HID to send with the data.

To prevent this situation, an ACCEPT should not be propagated back to the previous-hop until the HID negotiation with the next-hops has been completed.

Although it is possible that the negotiation extends for an arbitrary length of time, we consider this to be very unlikely. Since the HID is only relevant across a single hop, we can estimate the probability that a randomly selected HID will conflict with the HID of an established stream. Consider a stream in which the hop from an ST agent to ten next-hop agents is through the multicast facility of a given network. Assume also that each of the next-hop agents participates in 1000 other streams, and that each has been created with a different HID. A randomly selected 16-bit HID will have a probability of greater than 85.9% of succeeding on the first try, 98.1% of succeeding on the second, and 99.8% of succeeding on the third. We therefore suggest that a 16-bit HID space is sufficiently large to support ST until better multicast HID selection procedures, e.g., HID servers, can be deployed.

An obvious way to select the HID is for the ST agents to use a random number generator as suggested above. An alternate mechanism is for the intermediate agents to use the HID contained in the incoming CONNECT message for all the outgoing CONNECT messages, and generate a random number only as a second choice. In this case, the origin ST agent would

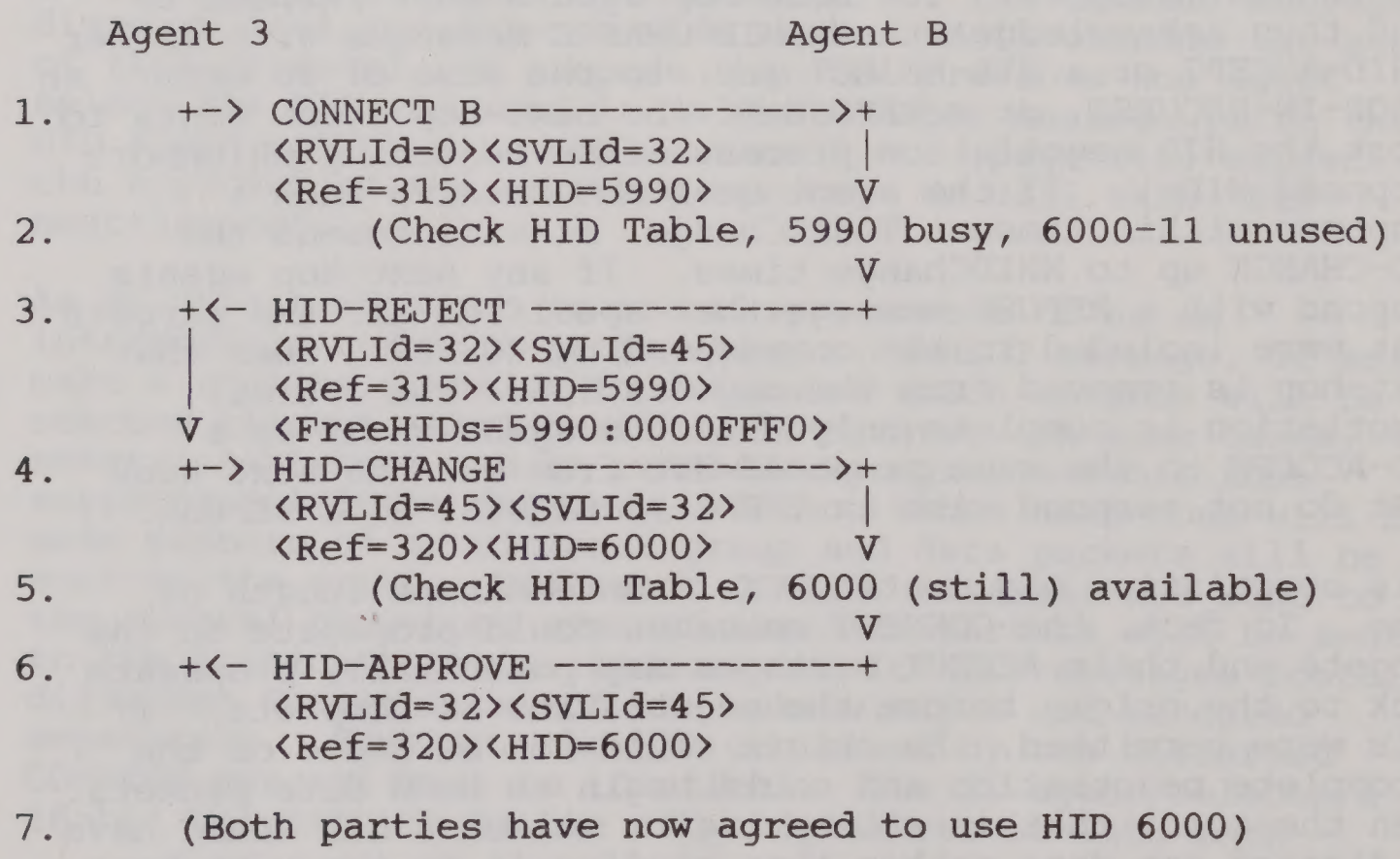


Figure 18. Typical HID Negotiation (No Multicasting)

be responsible for generating the HID, and the same HID could be propagated for the entire stream. This approach has the marginal advantage that the HID could be created by a higher layer protocol that might have global knowledge and could select small, globally unique HIDs for all the streams. While this is possible, we leave it for further study.

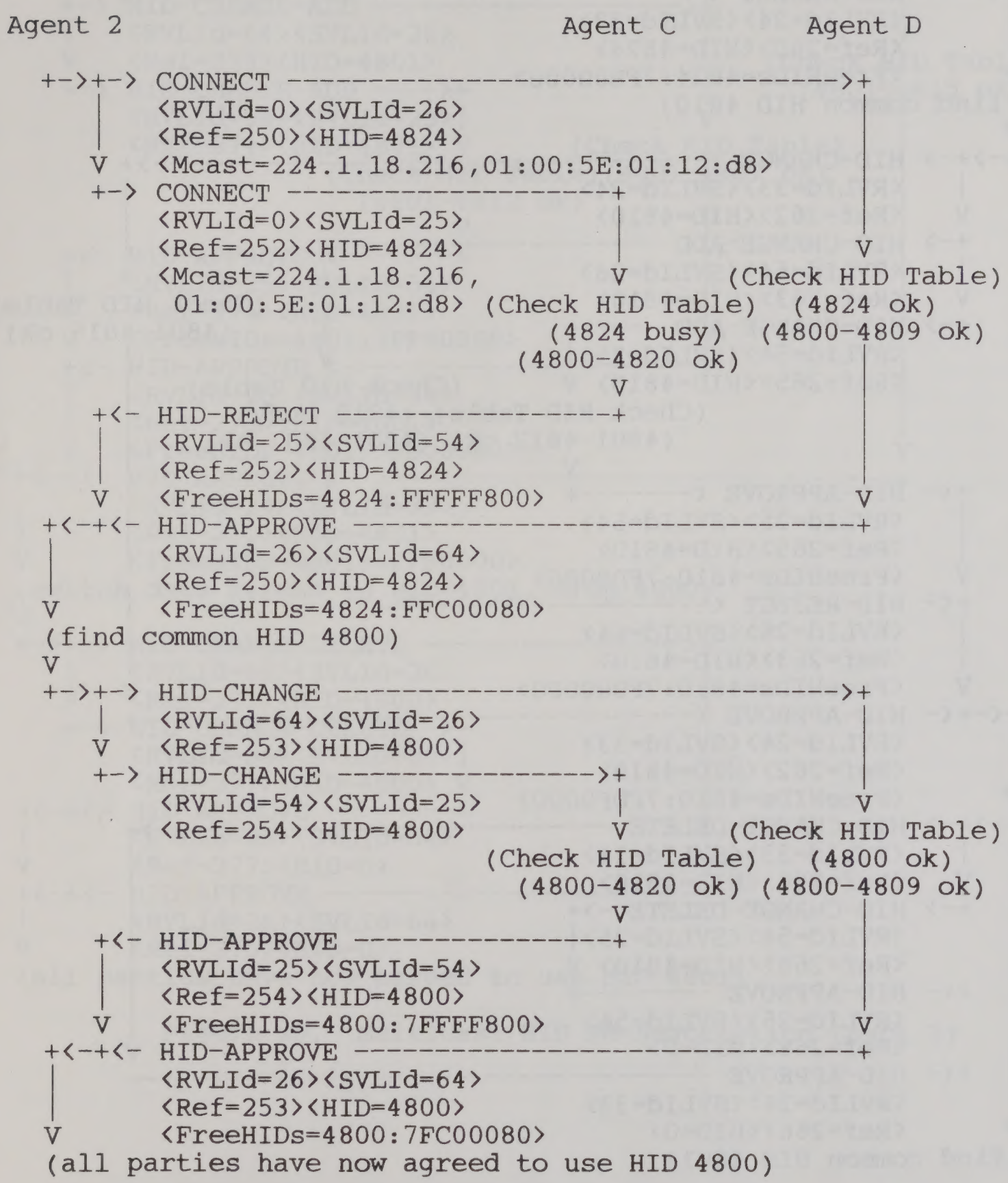


Figure 19. Multicast HID Negotiation

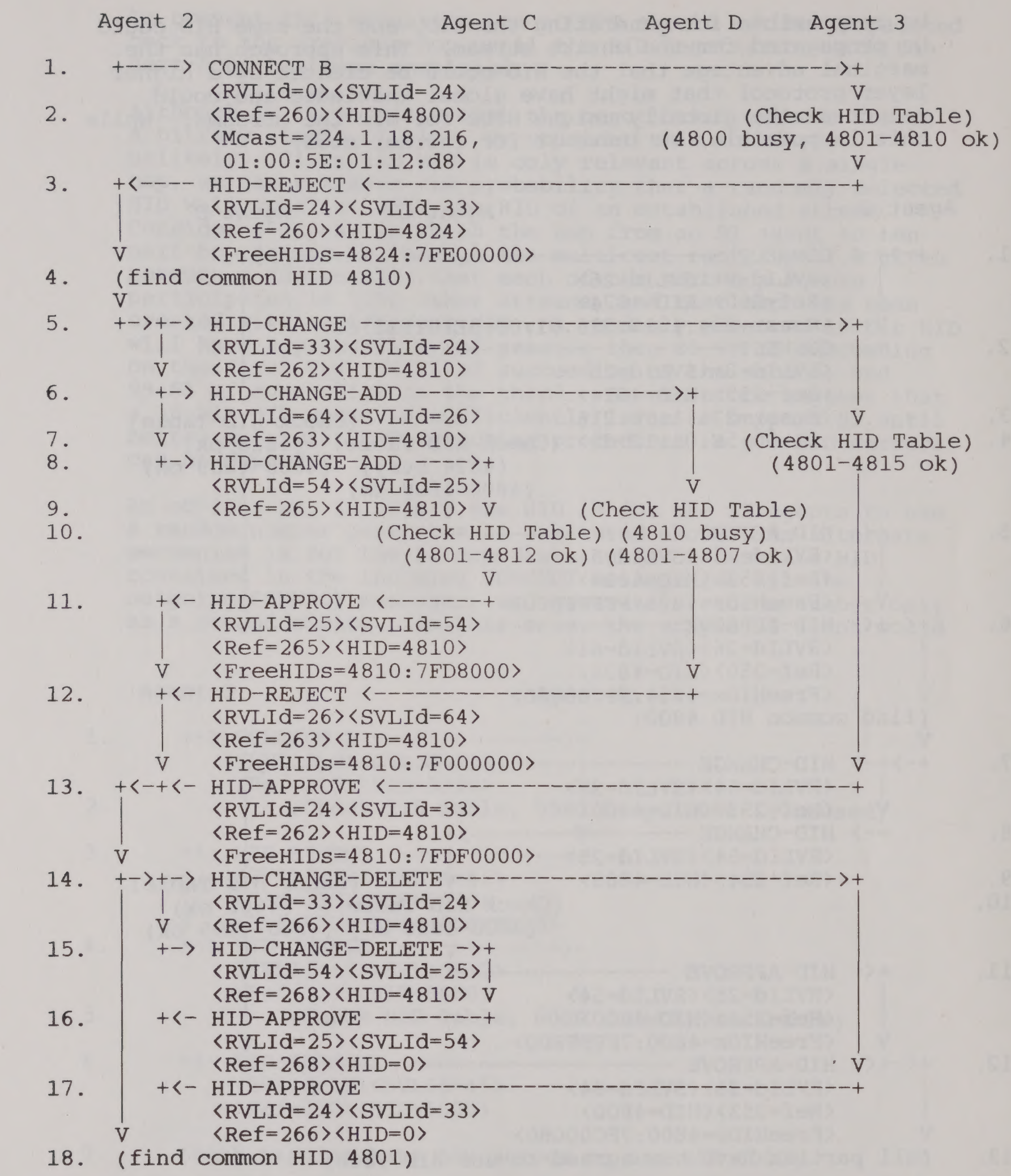


Figure 20. Multicast HID Re-Negotiation (part 1)

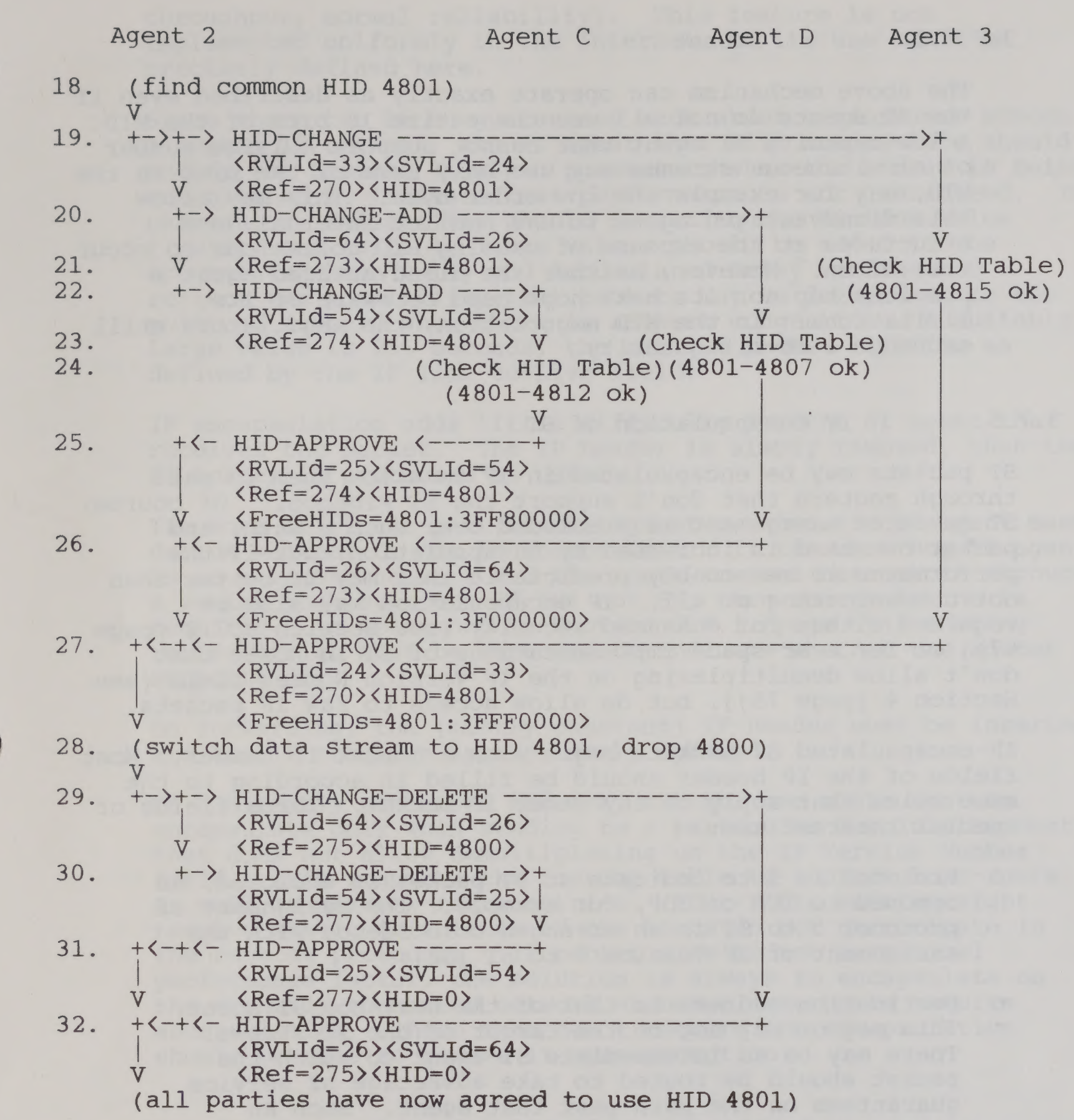


Figure 20. Multicast HID Re-Negotiation (part 2)

3.7.4.1. Subset

The above mechanism can operate exactly as described even if the ST agents do not all use the entire 16 bits of the HID. A low capacity ST agent that cannot support a large number of simultaneous streams may use only some of the bits in the HID, say for example the low order byte. This may allow this disadvantaged agent to use smaller internal data structures at the expense of causing HID collisions to occur more often. However, neither the disadvantaged agent's previous-hop nor its next-hops need be aware of its limitations. In the HID negotiation, the negotiators still exchange a 16-bit quantity.

3.7.5. IP Encapsulation of ST

ST packets may be encapsulated in IP to allow them to pass through routers that don't support the ST Protocol. Of course, ST resource management is precluded over such a path, and packet overhead is increased by encapsulation, but if the performance is reasonably predictable this may be better than not communicating at all. IP encapsulation may also be required either for enhanced security (see Section 3.7.8 (page 67)) or for user-space implementations of ST in hosts that don't allow demultiplexing on the IP Version Number field (see Section 4 (page 75)), but do allow access to raw IP packets.

IP-encapsulated ST packets begin with a normal IP header. Most fields of the IP header should be filled in according to the same rules that apply to any other IP packet. Three fields of special interest are:

- Protocol is 5 to indicate an ST packet is enclosed, as opposed to TCP or UDP, for example. The assignment of protocol 5 to ST is an arranged coincidence with the assignment of IP Version 5 to ST [18].

- Destination Address is that of the next-hop ST agent. This may or may not be the target of the ST stream. There may be an intermediate ST agent to which the packet should be routed to take advantage of service guarantees on the path past that agent. Such an intermediate agent would not be on a directly-connected network (or else IP encapsulation wouldn't be needed), so it would probably not be listed in the normal routing table. Additional routing mechanisms, not defined here, will be required to learn about such agents.

- Type-of-Service may be set to an appropriate value for the service being requested (usually low delay, high

throughput, normal reliability). This feature is not implemented uniformly in the Internet, so its use can't be precisely defined here.

Since there can be no guarantees made about performance across a normal IP network, the ST agent that will encapsulate should modify the Desired FlowSpec parameters when the stream is being established to indicate that performance is not guaranteed. In particular, Reliability should be set to the minimum value (1/256), and suitably large values should be added to the Accumulated Mean Delay and Accumulated Delay Variance to reflect the possibility that packets may be delayed up to the point of discard when there is network congestion. A suitably large value is 255 seconds, the maximum packet lifetime as defined by the IP Time-to-Live field.

IP encapsulation adds little difficulty for the ST agent that receives the packet. The IP header is simply removed, then the ST header is processed as usual.

The more difficult part is during setup, when the ST agent must decide whether or not to encapsulate. If the next-hop ST agent is on a remote network and the route to that network is through a router that supports IP but not ST, then encapsulation is required. As mentioned in Section 3.8.1 (page 69), routing table entries must be expanded to indicate whether the router supports ST.

On forwarding, the (mostly constant) IP Header must be inserted and the IP checksum appropriately updated.

On a directly connected network, though, one might want to encapsulate only when sending to a particular destination host that does not allow demultiplexing on the IP Version Number field. This requires the routing table to include host-route as well as network-route entries. Host-route entries might require static definition if the hosts do not participate in the routing protocols. If packet size is not a critical performance factor, one solution is always to encapsulate on the directly connected network whenever some hosts require encapsulation. Those that don't require the encapsulation should be able to remove it upon reception.

3.7.5.1. IP Multicasting

If an ST agent must use IP encapsulation to reach multiple next-hops toward different targets, then either the packet must be replicated for transmission to each next-hop, or IP multicasting [6] may be used if it is implemented in the next-hop ST agents and in the intervening IP routers.

This is analogous to using network-level service to multicast to several next-hop agents on a directly connected network.

When the stream is established, the collection of next-hop ST agents must be set up as an IP multicast group. It may be necessary for the ST agent that wishes to send the IP multicast to allocate a transient multicast group address and then tell the next-hop agents to join the group. Use of the MulticastAddress parameter (see Section 4.2.2.7 (page 86)) provides one way that the information may be communicated, but other techniques are possible. The multicast group address in inserted in the Destination Address field of the IP encapsulation when data packets are transmitted.

A block of transient IP multicast addresses, 224.1.0.0 - 224.1.255.255, has been allocated for this purpose. There are 2^16 addresses in this block, allowing a direct mapping with 16-bit HIDs, if appropriate. The mechanisms for allocating these addresses are not defined here.

In addition, two permanent IP multicast addresses have been assigned to facilitate experimentation with exchange of routing or other information among ST agents. Those addresses are:

224.0.0.7 All ST routers
224.0.0.8 All ST hosts

An ST router is an ST agent that can pass traffic between attached networks; an ST host is an ST agent that is connected to a single network or is not permitted to pass traffic between attached networks. Note that the range of these multicasts is normally just the attached local network, limited by setting the IP time-to-live field to 1 (see [6]).

3.7.6. Retransmission

The ST Control Message Protocol is made reliable through use of retransmission when an expected acknowledgment is not received in a timely manner. The problem of when to send a retransmission has been studied for protocols such as TCP [2] [10] [11]. The problem should be simpler for ST since control messages usually only have to travel a single hop and they do not contain very much data. However, the algorithms developed for TCP are sufficiently simple that their use is recommended for ST as well; see [2]. An implementor might, for example, choose to keep statistics separately for each

neighboring ST agent, or combined into a single statistic for an attached network.

Estimating the packet round-trip time (RTT) is a key function in reliable transport protocols such as TCP. Estimation must be dynamic, since congestion and resource contention result in varying delays. If RTT estimates are too low, packets will be retransmitted too frequently, wasting network capacity. If RTT estimates are too high, retransmissions will be delayed reducing network throughput when transmission errors occur. Article [11] identifies problems that arise when RTT estimates are poor, outlines how RTT is used and how retransmission timeouts (RTO) are estimated, and surveys several ways that RTT and RTO estimates can be improved.

Note the HELLO/ACK mechanism described in Section 3.7.1.2 (page 49) can give an estimate of the RTT and its variance. These estimates are also important for use with the delay and delay variance entries in the FlowSpec.

3.7.7. Routing

ST requires access to routing information in order to select a path from an origin to the destination(s). However, routing is considered to be a separate issue and neither the routing algorithm nor its implementation is specified here. ST should operate equally well with any reasonable routing algorithm.

While ST may be capable of using several types of information that are not currently available, the minimal information required is that provided by IP, namely the ability to find an interface and next hop router for a specified IP destination address and Type of Service. Methods to make more information available and to use it are left for further study. For initial ST implementations, any routing information that is required but not automatically provided will be assumed to be manually configured into the ST agents.

3.7.8. Security

The ST Protocol by itself does not provide security services. It is more vulnerable to misdelivery and denial of service than IP since the ST Header only carries a 16-bit HID for identification purposes. Any information, such as source and destination addresses, which a higher-layer protocol might use to detect misdelivery are the responsibility of either the application or higher-layer protocol.

ST is less prone to traffic analysis than IP since the only identifying information contained in the ST Header is a hop-by-hop identifier (HID). However, the use of a HID is also what makes ST more vulnerable to denial of service since an ST agent has no reliable way to detect when bogus traffic is injected into, and thus consumes bandwidth from, a user's stream. Detection can be enhanced through use of per-interface forwarding tables and verification of local network source and destination addresses.

We envision that applications that require security services will use facilities, such as the Secure Digital Networking System (SDNS) layer 3 Security Protocol (SP3/D) [19] [20]. In such an environment, ST PDUs would first be encapsulated in an IP Header, using IP Protocol 5 (ST) as described in Section 3.7.5 (page 64). These IP datagrams would then be secured using SP3/D, which results in another IP Protocol 5 PDU that can be passed between ST agents.

This memo does not specify how an application invokes security services.

3.8. ST Service Interfaces

ST has several interfaces to other modules in a communication system. ST provides its services to applications or transport-level protocols through its "upper" interface (or SAP). ST in turn uses the services provided by network layers, management functions (e.g., address translation and routing), and IP. The interfaces to these modules are described in this section in the form of subroutine calls. Note that this does not mean that an implementation must actually be implemented as subroutines, but is instead intended to identify the information to be passed between the modules.

In this style of outlining the module interfaces, the information passed into a module is shown as arguments to the subroutine call. Return information and/or success/failure indications are listed after the arrow ("->") that follows the subroutine call. In several cases, a list of values must either be passed to or returned from a module interface. Examples include a set of target addresses, or the mappings from a target list to a set of next hop addresses that span the route to the originally listed targets. When such a list is appropriate, the values repeated for each list element are bracketed and an asterisk is added to indicate that zero, one, or many list elements can be passed across the interface (e.g., "<target>*" means zero, one, or more targets).

3.8.1. Access to Routing Information

The design of routing functions that can support a variety of resource management algorithms is difficult. In this section we suggest a set of preliminary interfaces suitable for use in initial experiments. We expect that these interfaces will change as we gain more insight into how routing, resource allocation, and decision making elements are best divided.

Routing functions are required to identify the set of potential routes to each destination site. The routing functions should make some effort to identify routes that are currently available and that meet the resource requirements. However, these properties need not be confirmed until the actual resource allocation and connection setup propagation are performed.

The minimum capability required of the interface to routing is to identify the network interface and next hop toward a given target. We expect that the traditional routing table will need to be extended to include information that ST requires such as whether or not a next hop supports ST, and, if so, whether or not IP encapsulation (see Section 3.7.5 (page 64)) is required to communicate with it. In particular, host entries will be required for hosts that can only support ST through encapsulation because the IP software either is not capable of demultiplexing datagrams based on the IP Version Number field, or the application interface only supports access to raw IP datagrams. This interface is illustrated by the function:

```
FindNextHop( destination, TOS )
   -> result, < interface, next hop, ST-capable,
      MustEncapsulate >*
```

However, the resource management functions can best tradeoff among alternative routes when presented with a matrix of all potential routes. The matrix entry corresponding to a destination and a next hop would contain the estimated characteristics of the corresponding pathway. Using this representation, the resource management functions can quickly determine the next hop sets that cover the entire destination list, and compare the various parameters of the tradeoff between the guarantees that can be promised by each set. An interface that returns a compressed matrix, listing the suitable routes by next hop and the destinations reachable through each, is illustrated by the function:

```
FindNextHops( < destination >*, TOS )
   -> result, < destination, < interface, next hop,
      ST-capable, MustEncapsulate >* >*
```

We hope that routing protocols will be available that propagate additional metrics of bandwidth, delay, bit/burst error rate, and whether a router has ST capability. However, propagating this information in a timely fashion is still a key research issue.

3.8.2. Access to Network Layer Resource Reservation

The resources required to reach the next-hops associated with the chosen routes must be allocated. These allocations will generally be requested and released incrementally. As the next-hop elements for the routes are chosen, the network resources between the current node and the next-hops must be allocated. Since the resources are not guaranteed to be available -- a network or node further down the path might have failed or needed resources might have been allocated since the routing decisions where made -- some of these allocations may have to be released, another route selected, and a new allocation requested.

There are four basic interface functions needed for the network resource allocator. The first checks to see if the required resources are available, returning the likelihood that an ensuing resource allocation will succeed. A probability of 0% indicates the resources are not available or cannot promise to meet the required guarantees. Low probabilities indicate that most of the resource has been allocated or that there is a lot of contention for using the resource. This call does not actually reserve the resources:

```
ResourceProbe( requirements )
   -> likelihood
```

Another call reserves the resources:

```
ResourceReserve( requirements )
   -> result, reservation_id
```

The third call adjusts the resource guarantees:

```
ResourceAdjust( reservation_id, new requirements )
   -> result
```

The final call allows the resources to be released:

```
ResourceRelease( reservation_id )
   -> result
```

3.8.3. Network Layer Services Utilized

ST requires access to the usual network layer functions to send and receive packets and to be informed of network status information. In addition, it requires functions to enable and disable reception of multicast packets. Such functions might be defined as:

```
JoinLocalGroup( network level group-address )
   -> result, multicast_id

LeaveLocalGroup( network level group-address )
   -> result

RecvNet( SAP )
   -> result, src, dst, len, BufPTR )

SendNet( src, dst, SAP, len, BufPTR )
   -> result

GetNotification( SAP )
   -> result, infop
```

3.8.4. IP Services Utilized

Since ST packets might be sent or received using IP encapsulation, IP level routines to join and leave multicast groups are required in addition to the usual services defined in the IP specification (see the IP specification [2] [15] and the IP multicast specification [6] for details).

```
JoinHostGroup( IP level group-address, interface )
   -> result, multicast_id

LeaveHostGroup( IP level group-address, interface )
   -> result

GET_SRCADDR( remote IP addr, TOS )
   -> local IP address

SEND( src, dst, prot, TOS, TTL, BufPTR, len, Id, DF,
      opt )
   -> result

RECV( BufPTR, prot )
   -> result, src, dst, SpecDest, TOS, len, opt

GET_MAXSIZES( local, remote, TOS )
   -> MMS_R, MMS_S
```

```
ADVISE_DELIVPROB( problem, local, remote, TOS )
   -> result

SEND_ICMP( src, dst, TOS, TTL, BufPTR, len, Id, DF, opt )
   -> result

RECV_ICMP( BufPTR )
   -> result, src, dst, len, opt
```

3.8.5. ST Layer Services Provided

Interface to the ST layer services may be modeled using a set of subroutine calls (but need not be implemented as such). When the protocol is implemented as part of an operating system, these subroutines may be used directly by a higher level protocol processing layer.

These subroutines might also be provided through system service calls to provide a raw interface for use by an application. Often, this will require further adaptation to conform with the idiom of the particular operating system. For example, 4.3 BSD UNIX (TM) provides sockets, ioctls and signals for network programming.

```
open( connect/listen, SAPBytes, local SAP, local host,
      account, authentication info, < foreign host,
      SAPBytes, foreign SAP, options >*, flow spec,
      precedence, group name, optional parameters )
   -> result, id, stream name, < foreign host,
      foreign SAPBytes, foreign SAP, result, flow spec,
      rname, optional parameters >*
```

Note that an open by a target in "listen mode" may cause ST to create a state block for the stream to facilitate rendezvous.

```
add( id, SAPBytes, local SAP, local host, < foreign host,
     SAPBytes, foreign SAP, options >*, flow spec,
     precedence, group name, optional parameters )
   -> result, < foreign host, foreign SAPBytes,
      foreign SAP, result,
      flow spec, rname, optional parameters >*

send( id, buffer address, byte count, priority )
   -> result, next send time, burst send time

recv( id, buffer address, max byte count )
   -> result, byte count

recvsignal( id )
   -> result, signal, info
```

```
receivecontrol( id )
   -> result, id, stream name, < foreign host,
      foreign SAPBytes, foreign SAP, result, flow spec,
      rname, optional parameters >*

sendcontrol( id, flow spec, precedence, options,
      < foreign host, SAPBytes, foreign SAP, options >*)
   -> result, < foreign host, foreign SAPBytes,
      foreign SAP, result, flow spec, rname,
      optional parameters >*

change( id, flow spec, precedence, options,
      < foreign host, SAPBytes, foreign SAP, options >*)
   -> result, < foreign host, foreign SAPBytes,
      foreign SAP, result, flow spec, rname,
      optional parameters >*

close( id, < foreign host, SAPBytes, foreign SAP >*,
      optional parameters )
   -> result

status( id/stream name/group name )
   -> result, account, group name, protocol,
      < stream name, < foreign host, SAPbytes,
      foreign SAP, state, options, flow spec,
      routing info, rname >*, precedence, options >*

creategroup( members* )
   -> result, group name

deletegroup( group name, members* )
   -> result
```

[This page intentionally left blank.]

4. ST Protocol Data Unit Descriptions

The ST PDUs sent between ST agents consist of an ST Header ncapsulating either a higher layer PDU or an ST Control Message. Since ST operates as an extension of IP, the packet arrives at the same network service access point that IP uses to receive IP datagrams, e.g., ST would use the same ethertype (0x800) as does IP. The two types of packets are distinguished by the IP Version Number field (the first four bits of the packet); IP currently uses a value of 4, while ST has been assigned the value 5 [18]. There is no requirement for compatibility between IP and ST packet headers beyond the first four bits.

The ST Header also includes an ST Version Number, a total length field, a header checksum, and a HID, as shown in Figure 21. See Appendix 1 (page 147) for an explanation of the notation.

ST is the IP Version Number assigned to identify ST packets. The value for ST is 5.

Ver is the ST Version Number. This document defines ST Version 2.

Pri is the priority of the packet. It is used in data packets to indicate those packets to drop if a stream is exceeding its allocation. Zero is the lowest priority and 7 the highest.

T (bit 11) is used to indicate that a Timestamp is present following the ST Header but before any next higher layer protocol data. The Timestamp is not permitted on ST Control Messages (which may use the OriginTimestamp option).

Bits 12 through 15 are spares and should be set to 0.

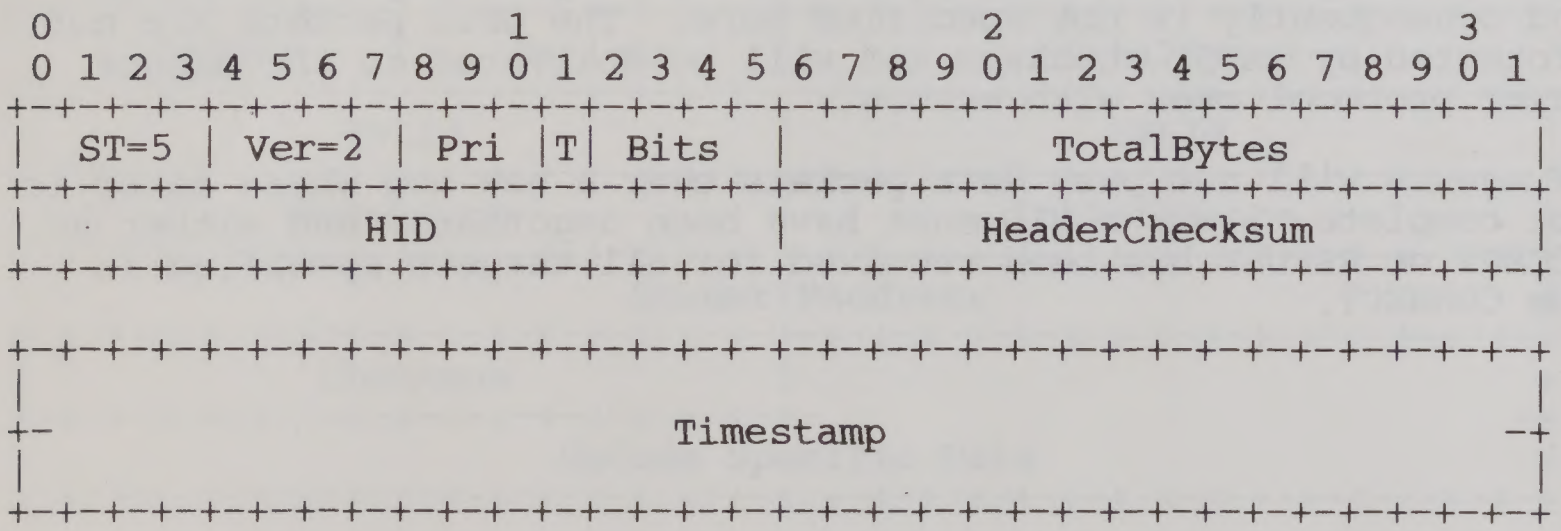


Figure 21. ST Header

TotalBytes is the length, in bytes, of the entire ST packet, it includes the ST Header and optional Timestamp but does not include any local network headers or trailers. In general, all length fields in the ST Protocol are in units of bytes.

HID is the 16-bit hop-by-hop stream identifier. It is an abbreviation for the Name of the stream and is used both to reduce the packet header length and, by the receiver of the data packet, to make the forwarding function more efficient. Control Messages have a HID value of zero. HIDs are negotiated by the next-hop and previous-hop agents to make the abbreviation unique. It is used here in the ST Header and in various Control Messages. HID values 1-3 are reserved for future use.

HeaderChecksum covers only the ST Header and Timestamp, if present. The ST Protocol uses 16-bit checksums here in the ST Header and in each Control Message. The standard Internet checksum algorithm is used: "The checksum field is the 16-bit one's complement of the one's complement sum of all 16-bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero." See [1] [12] [15] for suggestions for efficient checksum algorithms.

Timestamp is an optional timestamp inserted into data packets by the origin. It is only present when the T bit, described above, is set (1). Its use is negotiated at connection setup time; see Sections 4.2.3.5 (page 108) and 4.2.3.1 (page 100). The Timestamp has the NTP format; see [13].

4.1. Data Packets

ST packets whose HID is not zero to three are user data packets. Their interpretation is a matter for the higher layer protocols and consequently is not specified here. The data packets are not protected by an ST checksum and will be delivered to the higher layer protocol even with errors.

ST agents will not pass data packets over a new hop whose setup is not complete, i.e., a HID must have been negotiated and either an ACCEPT or REFUSE has been received for all targets specified in the CONNECT.

4.2. ST Control Message Protocol Descriptions

ST Control Messages are between a previous-hop agent and its next-hop agent(s) using a HID of zero. The control protocol follows a request-response model with all requests expecting responses. Retransmission after timeout (see Section 3.7.6 (page 66)) is used to allow for lost or ignored messages. Control messages do not extend across packet boundaries; if a control message is too large for the MTU of a hop, its information (usually a TargetList) is partitioned and a control message per partition is sent. All control messages have the following format:

OpCode identifies the type of control message. Each is described in detail in following sections.

Options is used to convey OpCode-specific variations for a control message.

TotalBytes is the length of the control message, in bytes, including all OpCode specific fields and optional parameters. The value is always divisible by four.

RVLId is used to convey the Virtual Link Identifier of the receiver of the control message, when known, or zero in the case of an initial CONNECT or diagnostic message. The RVLId is intended to permit efficient dispatch to the portion of a stream's state machine containing information about a specific operation in progress over the link. RVLId values 1-3 are reserved; see Sections 3 (page 17) and 3.7.1.2 (page 49).

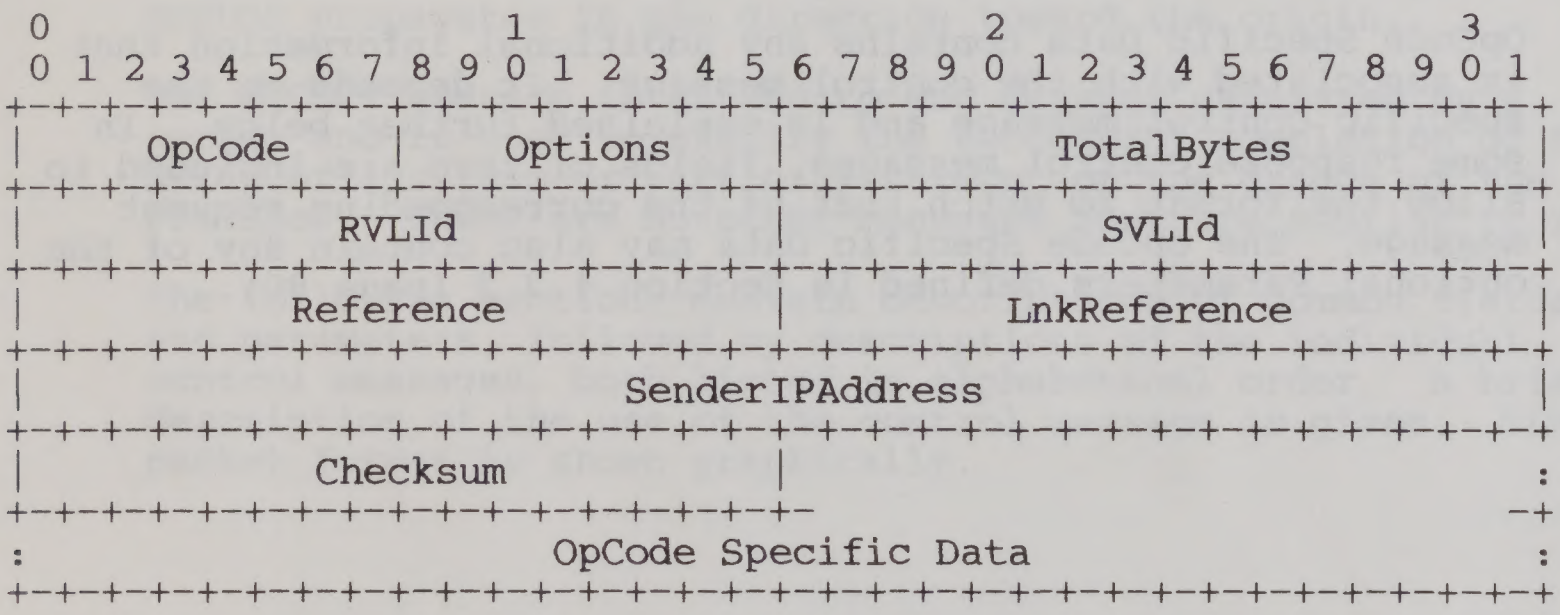

0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
OpCode
Options
TotalBytes
RVLId
SVLId
Reference
LnkReference
SenderIPAddress
Checksum
OpCode Specific Data

Figure 22. ST Control Message Format

SVLId is used to convey the Virtual Link Identifier of the sender of the control message. Except for ERROR-IN-REQUEST and diagnostic messages, it must never be zero. SVLId values 1-3 are reserved; see Sections 3 (page 17) and 3.7.1.2 (page 49).

Reference is a transaction number. Each sender of a request control message assigns a Reference number to the message that is unique with respect to the stream. The Reference number is used by the receiver to detect and discard duplicates. Each acknowledgment carries the Reference number of the request being acknowledged. Reference zero is never used, and Reference numbers are assumed to be monotonically increasing with wraparound so that the older-than and more-recent-than relations are well defined.

LnkReference contains the Reference field of the request control message that caused this request control message to be created. It is used in situations where a single request leads to multiple "responses". Examples are CONNECT and CHANGE messages that must be acknowledged hop-by-hop and will also lead to an ACCEPT or REFUSE from each target in the TargetList.

SenderIPAddress is the 32-bit IP address of the network interface that the ST agent used to send the control message. This value changes each time the packet is forwarded by an ST agent (hop-by-hop).

Checksum is the checksum of the control message. Because the control messages are sent in packets that may be delivered with bits in error, each control message must be checked before it is acted upon; see Section 4 (page 76).

OpCode Specific Data contains any additional information that is associated with the control message. It depends on the specific control message and is explained further below. In some response control messages, fields of zero are included to allow the format to match that of the corresponding request message. The OpCode Specific Data may also contain any of the optional Parameters defined in Section 4.2.2 (page 80).

4.2.1. ST Control Messages

The CONNECT and CHANGE messages are used to establish or modify branches in the stream. They propagate in the direction from the origin toward the targets. They are end-to-end messages created by the origin. They propagate all the way to the targets, and require ERROR-IN-REQUEST, ACK, HID-REJECT, HID-APPROVE, ACCEPT, or REFUSE messages in response. The CONNECT message is the stream setup message. The CHANGE message is used to change the characteristics of an established stream. The CONNECT message is also used to add one or more targets to an existing stream and during recovery of a broken stream. Both messages have a TargetList parameter and are processed similarly.

The DISCONNECT message is used to tear down streams or parts of streams. It propagates in the direction from the origin toward the targets. It is either used as an end-to-end message generated by the origin that is used to completely tear down a stream, or is generated by an intermediate ST agent that preempts a stream or detects the failure of its previous-hop agent or network in the stream. In the latter case, it is used to tear down the part of the stream from the failure to the targets, thus the message propagates all the way to the targets.

The REFUSE message is sent by a target to refuse to join or remove itself from a stream; in these cases, it is an end-to-end message. An intermediate ST agent issues a REFUSE if it cannot find a route to a target, can only find a route to a target through the previous-hop, preempts a stream, or detects a failure in a next-hop ST agent or network. In all cases a REFUSE propagates in the direction toward the origin.

The ACCEPT message is an end-to-end message generated by a target and is used to signify the successful completion of the setup of a stream or part of a stream, or the change of the FlowSpec. There are no other messages that are similar to it.

The following sections contain descriptions of common fields and parameters, followed by descriptions of the individual control messages, both listed in alphabetical order. A brief description of the use of the control message is given. The packet format is shown graphically.

4.2.2. Common SCMP Elements

Several fields and parameters (referred to generically as "elements") are common to two or more PDUs. They are described in detail here instead of repeating their description several times. In many cases, the presence of a parameter is optional. To permit the parameters to be easily defined and parsed, each is identified with a PCode byte that is followed by a PBytes byte indicating the length of the parameter in bytes (including the PCode, PByte, and any padding bytes). If the length of the information is not a multiple of 4 bytes, the parameter is padded with one to three zero (0) bytes. PBytes is thus always a multiple of four. Parameters can be present in any order.

4.2.2.1. DetectorIPAddress

Several control messages contain the DetectorIPAddress field. It is used to identify the agent that caused the first instance of the message to be generated, i.e., before it was propagated. It is copied from the received message into the copy of the message that is to be propagated to a previous-hop or next-hop. It use is primarily diagnostic.

4.2.2.2. ErroredPDU

The ErroredPDU parameter (PCode = 1) is used for diagnostic purposes to encapsulate a received ST PDU that contained an error. It may be included in the ERROR-IN-REQUEST, ERROR-IN-RESPONSE, or REFUSE messages. It use is primarily diagnostic.

PDUBytes indicates how many bytes of the PDUInError are actually present.

ErrorOffset contains the number of bytes into the errored PDU to the field containing the error. At least as much of the PDU in error must be included to

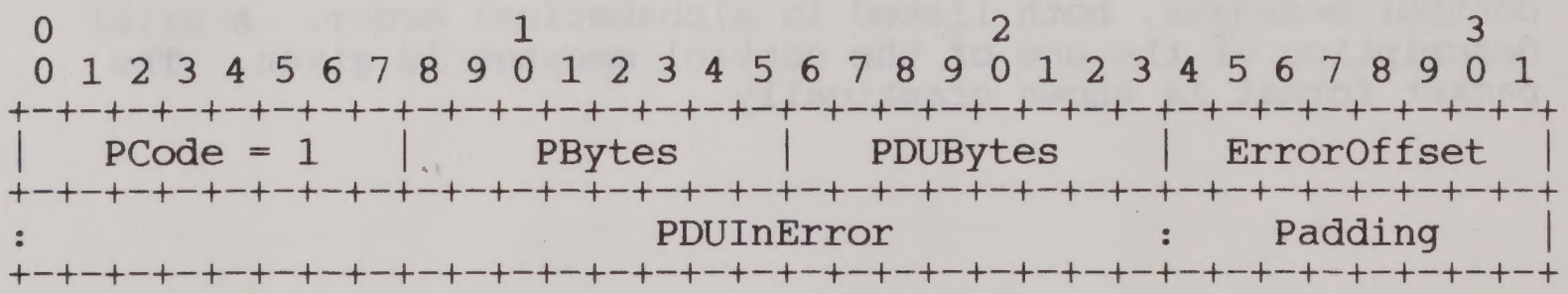


Figure 23. ErroredPDU

include the field or parameter identified by ErrorOffset; an ErrorOffset of zero would imply a problem with the IP Version Number or ST Version Number fields.

PDUInError is the PDU in error, beginning with the ST Header.

4.2.2.3. FlowSpec & RFlowSpec

The FlowSpec is used to convey stream service requirements end-to-end. We expect that other versions of FlowSpec will be needed in the future, which may or may not be subsets or supersets of the version described here. PBytes will allow new constraints to be added to the end without having to simultaneously update all implementations in the field. Implementations are expected to be able to process in a graceful manner a Version 4 (or higher) structure that has more elements than shown here.

The FlowSpec parameter (PCode = 2) is used in several messages to convey the FlowSpec.

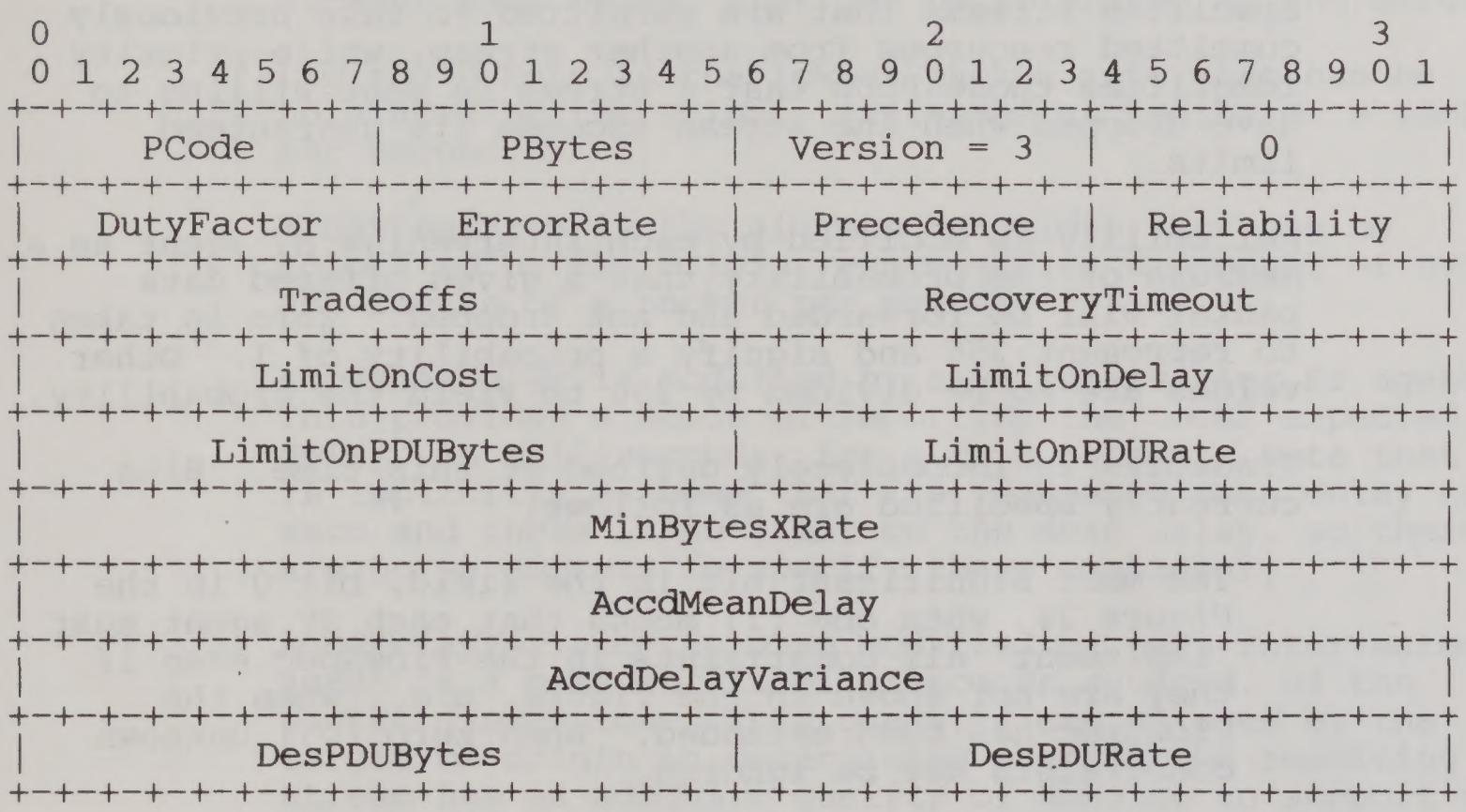


Figure 24. FlowSpec & RFlowSpec

The RFlowSpec parameter (PCode = 12) is used in conjunction with the FDx option to convey the FlowSpec that is to be used in the reverse direction.

Version identifies the version of the FlowSpec. Version 3 is defined here.

DutyFactor is the estimated proportion of the time that the requested bandwidth will actually be in use. Zero is taken to represent 256 and signify a duty factor of 1. Other values are to be divided by 256 to yield the duty factor.

ErrorRate expresses the error rate as the negative exponent of 10 in the error rate. One (1) represents a bit error rate of 0.1 and 10 represents 0.0000000001.

Precedence is the precedence of the connection being established. Zero represents the lowest precedence. Note that non-zero values of this parameter should be subject to authentication and authorization checks, which are not specified here. In general, the distinction between precedence and priority is that precedence specifies streams that are permitted to take previously committed resources from another stream, while priority identifies those PDUs that a stream is most willing to have dropped when the stream exceeds its guaranteed limits.

Reliability is modified by each intervening ST agent as a measure of the probability that a given offered data packet will be forwarded and not dropped. Zero is taken to represent 256 and signify a probability of 1. Other values are to be divided by 256 to yield the probability.

Tradeoffs is incompletely defined at this time. Bits currently specified are as follows:

The most significant bit in the field, bit 0 in the Figure 24, when one (1) means that each ST agent must "implement" all constraints in the FlowSpec even if they are not shown in the figure, e.g., when the FlowSpec has been extended. When zero (0), unknown constraints may be ignored.

The second most significant bit in the field, bit 1, when one (1) means that one or more constraints are unknown and have been ignored. When zero (0), all constraints are known and have been processed.

The third most significant bit in the field, bit 2, is used for RevChrg; see Section 3.6.5 (page 46).

Other bits are currently unspecified, and should be set to zero (0) by the origin ST agent and not changed by other agents unless those agents know their meaning.

RecoveryTimeout specifies the nominal number of milliseconds that the application is willing to wait for a failed system component to be detected and any corrective action to be taken.

LimitOnCost specifies the maximum cost that the origin is willing to expend. A value of zero indicates that the application is not willing to incur any direct charges for the resources used by the stream. The meaning of non-zero values is left for further study.

LimitOnDelay specifies the maximum end-to-end delay, in milliseconds, that can be tolerated by the origin.

LimitOnPDUBytes is the smallest packet size, in terms of ST-user data bytes, that can be tolerated by the origin.

LimitOnPDURate is the lowest packet rate that can be tolerated by the origin, expressed as tenths of a packet per second.

MinBytesXRate is the minimum bandwidth that can be tolerated by the origin, expressed as a product of bytes and tenths of a packet per second.

AccdMeanDelay is modified by each intervening ST agent. This provides a means of reporting the total expected delay, in milliseconds, for a data packet. Note that it is implicitly assumed that the requested mean delay is zero and there is no limit on the mean delay, so there are no parameters to specify these explicitly.

AccdDelayVariance is also modified by each intervening ST agent as a measure, in milliseconds squared, of the packet dispersion. This quantity can be used by the target or origin in determining whether the resulting stream has an adequate quality of service to support the application. Note that it is implicitly assumed that the requested delay variance is zero and there is no limit on the delay variance, so there are no parameters to specify these explicitly.

DesPDUBytes is the desired PDU size in bytes. This is not necessarily the same as the minimum necessary PDU size. This value may be made smaller by intervening ST agents so long as it is not made smaller than LimitOnPDUBytes. The *PDUBytes limits measure the size of the PDUs of next-higher protocol layer, i.e., the user information contained in a data packet. An ST agent must account for both the ST Header (including possible IP encapsulation) and any local network headers and trailers when comparing a network's MTU with *PDUBytes. In an ACCEPT message, the value of this field will be no larger than the MTU of the path to the specified target.

DesPDURate is the requested PDU rate, expressed as tenths of a packet per second. This value may be made smaller by intervening ST agents so long as it is not made smaller than LimitOnPDURate.

It is expected that the next parameter to be added to the FlowSpec will be a Burst Descriptor. This parameter will describe the burstiness of the offered traffic. For example, this may include the simple average rate, peak rate and variance values, or more complete descriptions that characterize the distribution of expected burst rates and their expected duration. The nature of the algorithms that deal with the traffic's burstiness and the information that needs to be described by this parameter will be subjects of further experimentation. It is expected that a new FlowSpec with Version = 4 will be defined that looks like Version 3 but has a Burst Descriptor parameter appended to the end.

4.2.2.4. FreeHIDs

The FreeHIDs parameter (PCode = 3) is used to communicate to the previous-hop suggestions for a HID. It consists of BaseHID and FreeHIDBitMask fields. Experiments will determine how long the mask should be for practical use of this parameter. The parameter (if implemented) should be included in all HID-REJECTs, and in HID-APPROVEs that are linked to a multicast CONNECT, e.g., one containing the MulticastAddress parameter.

BaseHID was the suggested value in a HID-CHANGE or CONNECT. BaseHID is chosen to be the suggested HID value to insure that the masks from multiple FreeHIDs parameters will overlap.

FreeHIDBitMask identifies available HID values as follows. Bit 0 in the FreeHIDBitMask corresponds to a

HID with a value equal to BaseHID with the 5 least significant bits set to zero, bit 1 corresponds to that value + 1, etc. This alignment of the mask on a 32-bit boundary is used so that masks from several FreeHIDs parameters might more easily be combined using a bit-wise AND function to find a free HID.

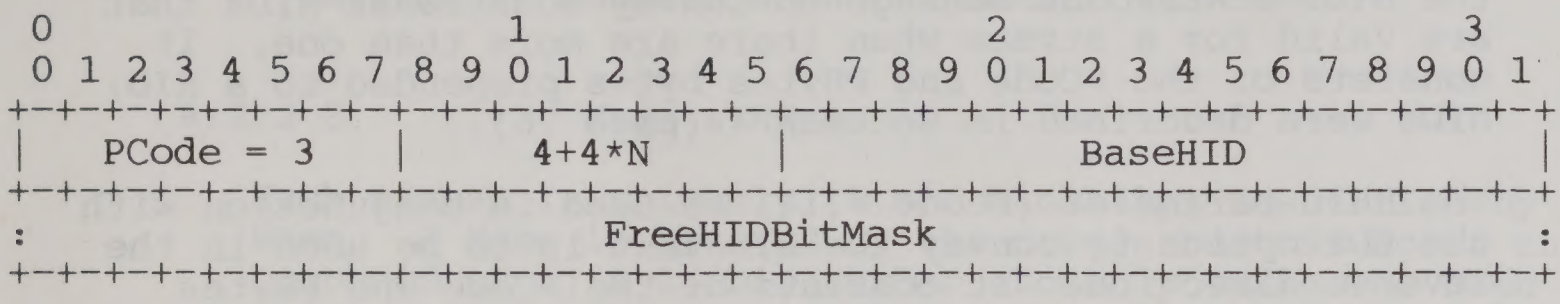


Figure 25. FreeHIDs

4.2.2.5. Group & RGroup

The Group parameter (PCode = 4) is an optional argument used only for the creation of a stream. This parameter contains a GroupName; the GroupName may be the same as the Name of one of the group's streams. In addition, there may be some number of <SubGroupId, Relation> tuples that describe the meaning of the grouping and the relation between the members of the group. The forms of grouping are for further study.

The RGroup parameter (PCode = 13) is an optional argument used only for the creation of a stream in the reverse direction that is a member of a Group; see the FDx option, Section 3.6.3 (page 45). This parameter has the same format as the Group parameter.

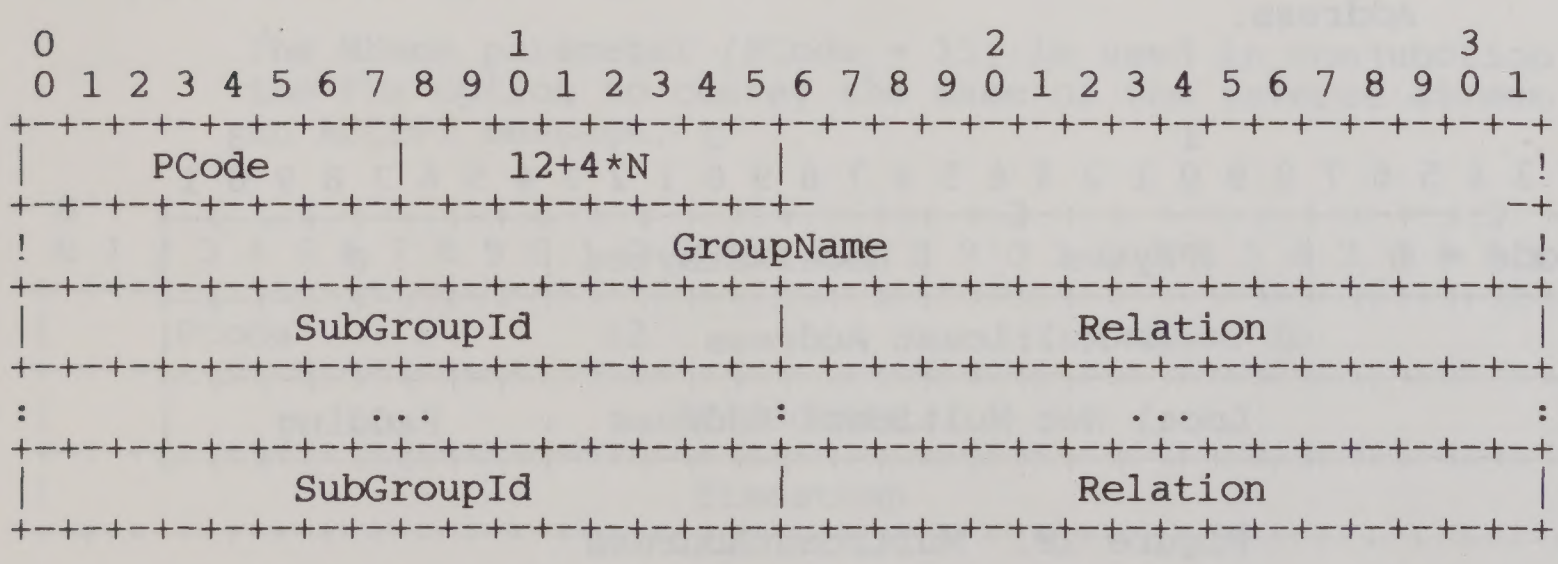


Figure 26. Group & RGroup

A GroupName has the same format as a Name; see Figure 29.

4.2.2.6. HID & RHID

The HID parameter (PCode = 5) is used in the NOTIFY message when the notification is related to a HID, and possibly in the STATUS-RESPONSE message to convey additional HIDs that are valid for a stream when there are more than one. It consists of the PCode and PBytes bytes prepended to a HID; HIDs were described in Section 4 (page 76).

The RHID parameter (PCode = 14) is used in conjunction with the FDx option to convey the HID that is to be used in the reverse direction. It consists of the PCode and PBytes bytes prepended to a HID.

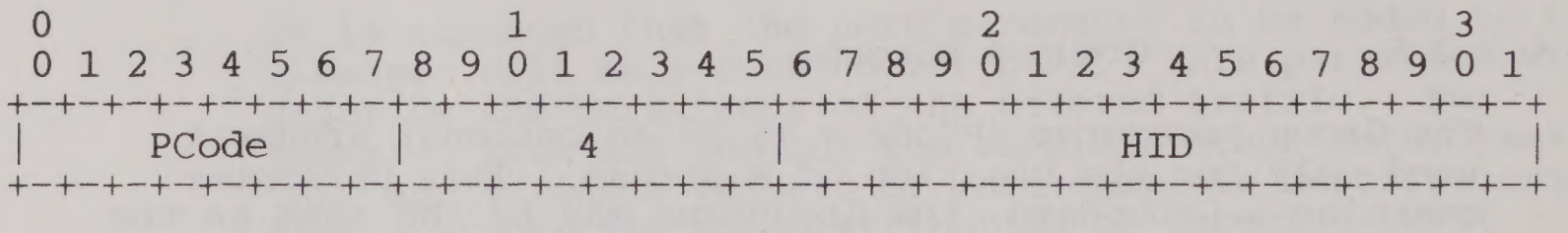


Figure 27. HID & RHID

4.2.2.7. MulticastAddress

The MulticastAddress parameter (PCode = 6) is an optional parameter that is used, when setting up a network level multicast group, to communicate an IP and/or local network multicast address to the next-hop agents that should become members of the group.

LocalNetBytes is the length of the Local Net Multicast Address.

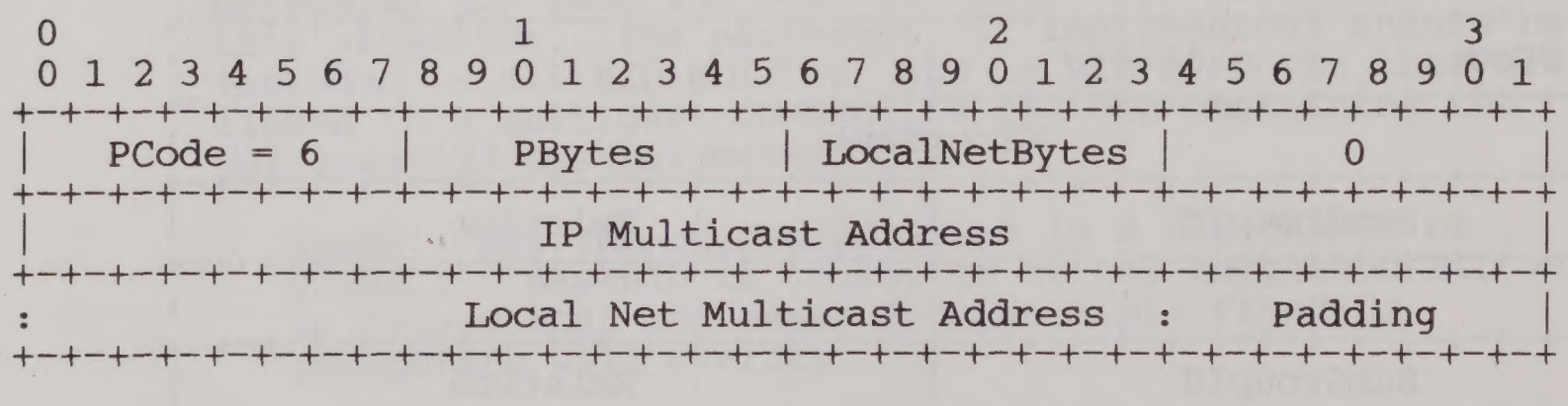

0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
PCode = 6
PBytes
LocalNetBytes
0
IP Multicast Address
Local Net Multicast Address
Padding

Figure 28. MulticastAddress

IP Multicast Address is described in [6]. This field is zero (0) if no IP multicast address is known or is applicable. The block of addresses 224.1.0.0 - 224.1.255.255 has been allocated for use by ST.

Local Net Multicast Address is the multicast address to be used on the local network. It corresponds to the IP Multicast Address when the latter is non-zero.

4.2.2.8. Name & RName

Each stream is uniquely (i.e., globally) identified by a Name. A Name is created by the origin host ST agent and is composed of 1) a 16-bit number chosen to make the Name unique within the agent, 2) the IP address of the origin ST agent, and 3) a 32-bit timestamp. If the origin has multiple IP addresses, then any that can be used to reach target may be used in the Name. The intent is that the <Unique ID, IP Address> tuple be unique for the lifetime of the stream. It is suggested that to increase robustness a Unique ID value not be reused for a period of time on the order of 5 minutes.

The Timestamp is included both to make the Name unique over long intervals (e.g., forever) for purposes of network management and accounting/billing, and to protect against failure of an ST agent that causes knowledge of active Unique IDs to be lost. The assumption is that all ST agents have access to some "clock". If this is not the case, the agent should have access to some form of non-volatile memory in which it can store some number that at least gets incremented per restart.

The Name parameter (PCode = 7) is used in most control messages to identify a stream.

The RName parameter (PCode = 15) is used in conjunction with the FDx option to convey the Name of the reverse stream in an ACCEPT message.

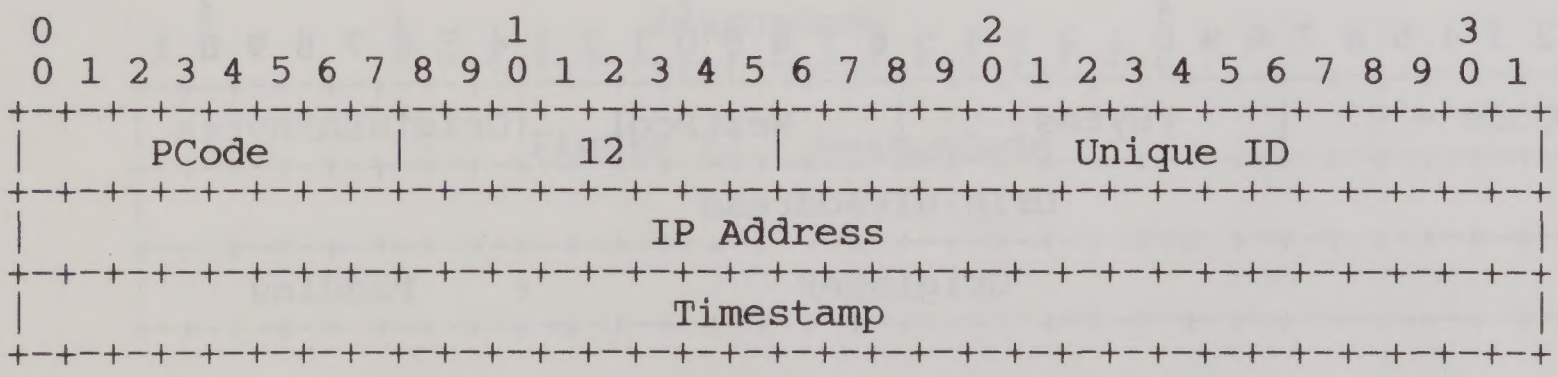


Figure 29. Name & RName

4.2.2.9. NextHopIPAddress

The NextHopIPAddress parameter (PCode = 8) is an optional parameter of NOTIFY (RouteBack) or REFUSE (RouteInconsist or RouteLoop) and contains the IP address of a suggested next-hop ST agent.

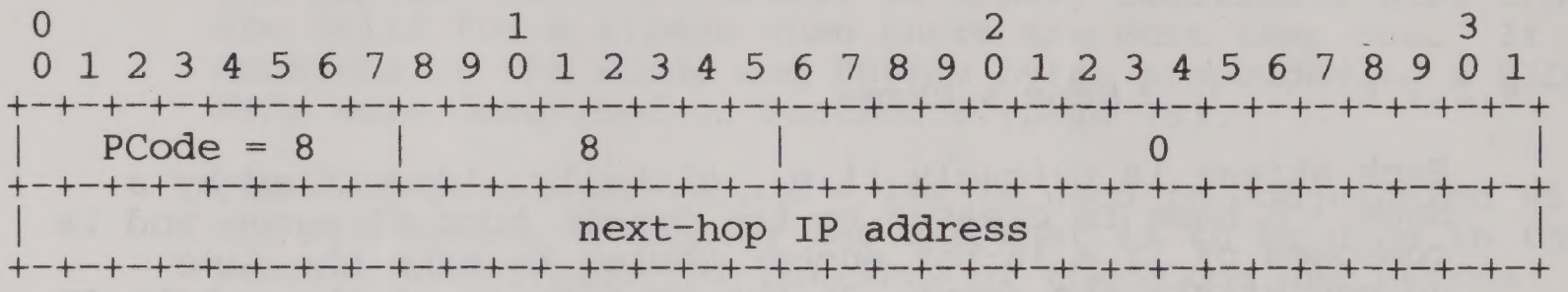


Figure 30. NextHopIPAddress

4.2.2.10. Origin

The Origin parameter (PCode = 9) is used to identify the origin of the stream, the next higher protocol, and the SAP being used in conjunction with that protocol.

NextPcol is an 8-bit field used in demultiplexing operations to identify the protocol to be used above ST. The values of NextPcol are in the same number space as the IP Header's Protocol field and are consequently defined in the Assigned Numbers RFC [18].

OriginSAPBytes specifies the length of the OriginSAP, exclusive of any padding required to maintain 32-bit alignment.

OriginIPAddress is (one of) the IP address of the origin.

OriginSAP identifies the origin's SAP associated with the NextPcol protocol.

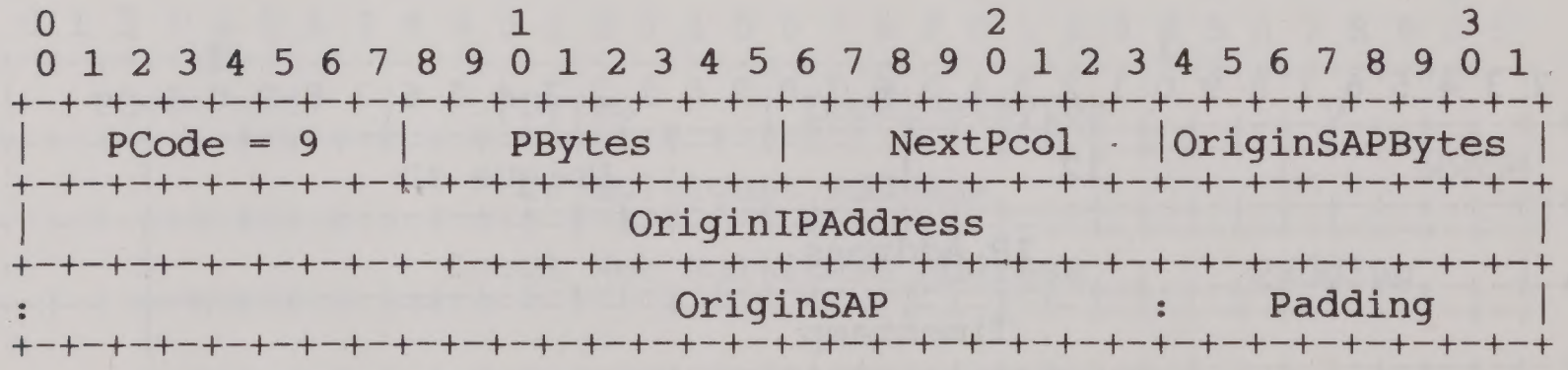

0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
PCode = 9
PBytes
NextPcol
OriginSAPBytes
OriginIPAddress
OriginSAP
Padding

Figure 31. Origin

4.2.2.11. OriginTimestamp

The OriginTimestamp parameter (PCode = 10) is used to indicate the time at which the control message was sent.

The units and format of the timestamp is that defined in the NTP protocol specification [13]. Note that discontinuities over leap seconds are expected.

Note that the time synchronization implied by the use of such a parameter is the subject of systems management functions not described in this memo, e.g., NTP.

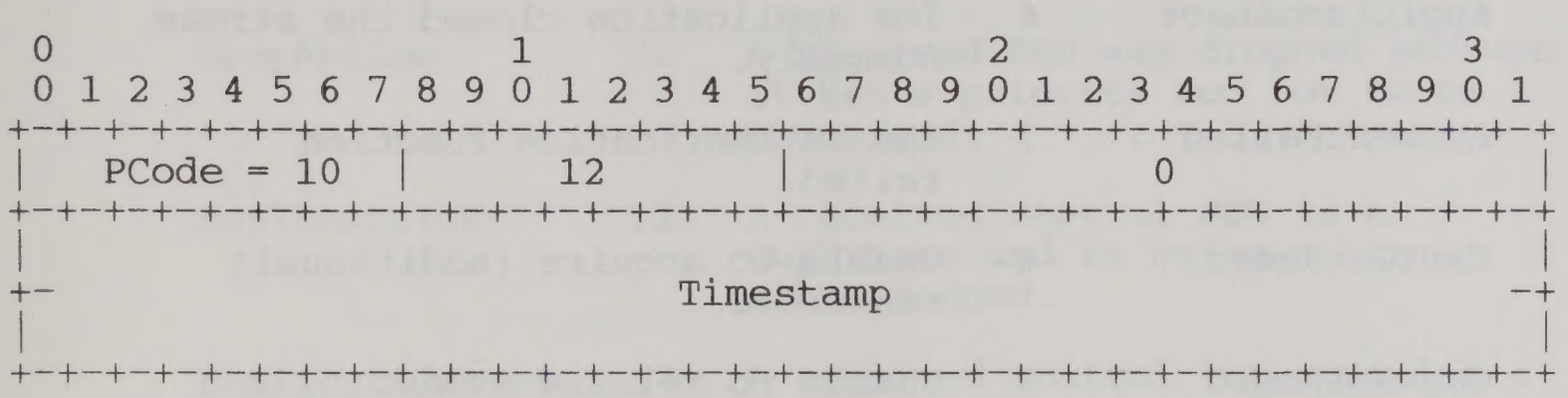


Figure 32. OriginTimestamp

4.2.2.12. ReasonCode

Several errors may occur during protocol processing. All ST error codes are taken from a single number space. The currently defined values and their meaning is presented in the list below. Note that new error codes may be defined from time to time. All implementations are expected to handle new codes in a graceful manner. If an unknown ReasonCode is encountered, it should be assumed to be fatal.

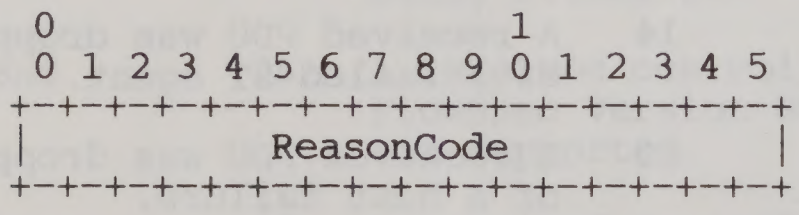


Figure 33. ReasonCode

Name	Value	Meaning
AcceptTimeout	2	An Accept has not been acknowledged.
AccessDenied	3	Access denied.
AckUnexpected	4	An unexpected ACK was received.
ApplAbort	5	The application aborted the stream abnormally.
ApplDisconnect	6	The application closed the stream normally.
AuthentFailed	7	The authentication function failed.
CantGetResrc	8	Unable to acquire (additional) resources.
CantRelResrc	9	Unable to release excess resources.
CksumBadCtl	10	A received control PDU has a bad message checksum.
CksumBadST	11	A received PDU has a bad ST Header checksum.
DropExcdDly	12	A received PDU was dropped because it could not be processed within the delay specification.
DropExcdMTU	13	A received PDU was dropped because its size exceeds the MTU.
DropFailAgt	14	A received PDU was dropped because of a failed ST agent.
DropFailHst	15	A received PDU was dropped because of a host failure.
DropFailIfc	16	A received PDU was dropped because of a broken interface.
DropFailNet	17	A received PDU was dropped because of a network failure.

Name	Value	Meaning
DropLimits	18	A received PDU was dropped because it exceeds the resource limits for its stream.
DropNoResrc	19	A received PDU was dropped due to no available resources (including precedence).
DropNoRoute	20	A received PDU was dropped because of no available route.
DropPriLow	21	A received PDU was dropped because it has a priority too low to be processed.
DuplicateIgn	22	A received control PDU is a duplicate and is being acknowledged.
DuplicateTarget	23	A received control PDU contains a duplicate target, or an attempt to add an existing target.
ErrorUnknown	1	An error not contained in this list has been detected.
failure	N/A	An abbreviation used in the text for any of the more specific errors: DropFailAgt, DropFailHst, DropFailIfc, DropFailNet, IntfcFailure, NetworkFailure, STAgentFailure, FailureRecovery.
FailureRecovery	24	A notification that recovery is being attempted.
FlowVerBad	25	A received control PDU has a FlowSpec Version Number that is not supported.
GroupUnknown	26	A received control PDU contains an unknown Group Name.
HIDNegFails	28	HID negotiation failed.
HIDUnknown	29	A received control PDU contains an unknown HID.

Name	Value	Meaning
InconsistHID	30	An inconsistency has been detected with a stream Name and corresponding HID.
InconsistGroup	31	An inconsistency has been detected with the streams forming a group.
IntfcFailure	32	A network interface failure has been detected.
InvalidHID	33	A received ST PDU contains an invalid HID.
InvalidSender	34	A received control PDU has an invalid SenderIPAddress field.
InvalidTotByt	35	A received control PDU has an invalid TotalBytes field.
LnkRefUnknown	36	A received control PDU contains an unknown LnkReference.
NameUnknown	37	A received control PDU contains an unknown stream Name.
NetworkFailure	38	A network failure has been detected.
NoError	0	No error has occurred.
NoRouteToAgent	39	Cannot find a route to an ST agent.
NoRouteToDest	40	Cannot find a route to the destination.
NoRouteToHost	41	Cannot find a route to a host.
NoRouteToNet	42	Cannot find a route to a network.
OpCodeUnknown	43	A received control PDU has an invalid OpCode field.
PCodeUnknown	44	A received control PDU has a parameter with an invalid PCode.
ParmValueBad	45	A received control PDU contains an invalid parameter value.

Name	Value	Meaning
PcolIdUnknown	46	A received control PDU contains an unknown next-higher layer protocol identifier.
ProtocolError	47	A protocol error was detected.
PTPError	48	Multiple targets were specified for a stream created with the PTP option.
RefUnknown	49	A received control PDU contains an unknown Reference.
RestartLocal	50	The local ST agent has recently restarted.
RemoteRestart	51	The remote ST agent has recently restarted.
RetransTimeout	52	An acknowledgment to a control message has not been received after several retransmissions.
RouteBack	53	The routing function indicates that the route to the next-hop is through the same interface as the previous-hop and is not the previous-hop.
RouteInconsist	54	A routing inconsistency has been detected, e.g., a route loop.
RouteLoop	55	A CONNECT was received that specified an existing target.
SAPUnknown	56	A received control PDU contains an unknown next-higher layer SAP (port).
STAgentFailure	57	An ST agent failure has been detected.
StreamExists	58	A stream with the given Name or HID already exists.
StreamPreempted	59	The stream has been preempted by one with a higher precedence.

Name	Value	Meaning
STVerBad	60	A received PDU is not ST Version 2.
TooManyHIDs	61	Attempt to add more HIDs to a stream than the implementation supports.
TruncatedCtl	62	A received control PDU is shorter than expected.
TruncatedPDU	63	A received ST PDU is shorter than the ST Header indicates.
UserDataSize	64	The UserData parameter is too large to permit a control message to fit into a network's MTU.

4.2.2.13. RecordRoute

The RecordRoute parameter (PCode = 11) may be used to request that the route between the origin and a target be recorded and returned to the agent specified in the DetectorIPAddress field.

FreeOffset is the offset to the position where the next next-hop IP address should be inserted. It is initialized to four (4) and incremented by four each time an agent inserts its IP address.

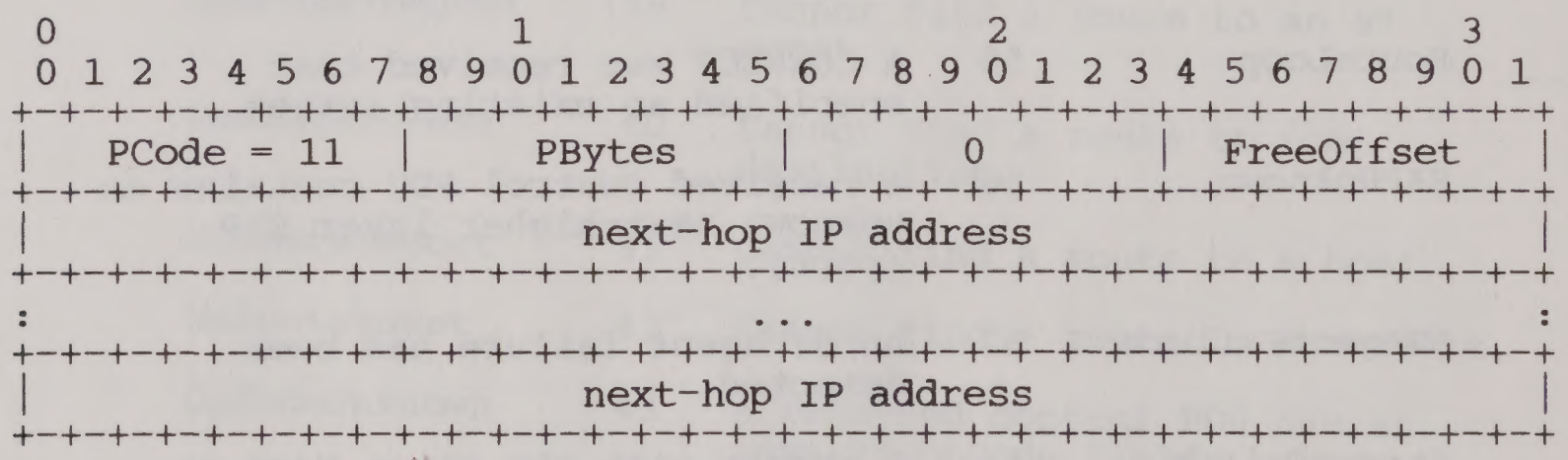


Figure 34. RecordRoute

4.2.2.14. SrcRoute

The SrcRoute parameter is used, in the Target structure shown in Figure 36, to specify the IP addresses of the ST agents through which the stream to the target should pass. There are two forms of the option, distinguished by the PCode.

With loose source route (PCode = 18) each ST agent first examines the first next-hop IP address in the option. If the address is (one of) the address of the current ST agent, that entry is removed, and the PBytes field reduced by four (4). If the resulting PBytes field contains 4 (i.e., there are no more next-hop IP addresses) the parameter is removed from the Target. In either case, the Target's TargetBytes field and the TargetList's PBytes field must be reduced accordingly. The ST agent then routes toward the first next-hop IP address in the option, if one exists, or toward the target otherwise. Note that the target's IP address is not included as the last entry in the list.

With a strict source route (PCode = 19) each ST agent first examines the first next-hop IP address in the option. If the address is not (one of) the address of the current ST agent, a routing error has occurred and should be reported with the appropriate reason code. Otherwise that entry is removed, and the PBytes field reduced by four (4). If the resulting PBytes field contains 4 (i.e., there are no more next-hop IP addresses) the parameter is removed from the Target. In either case, the Target's TargetBytes field and the TargetList's PBytes field must be reduced accordingly. The ST agent then routes toward the first next-hop IP address in the option, if one exists, or toward the target otherwise. Note that the target's IP address is not included as the last entry in the list.

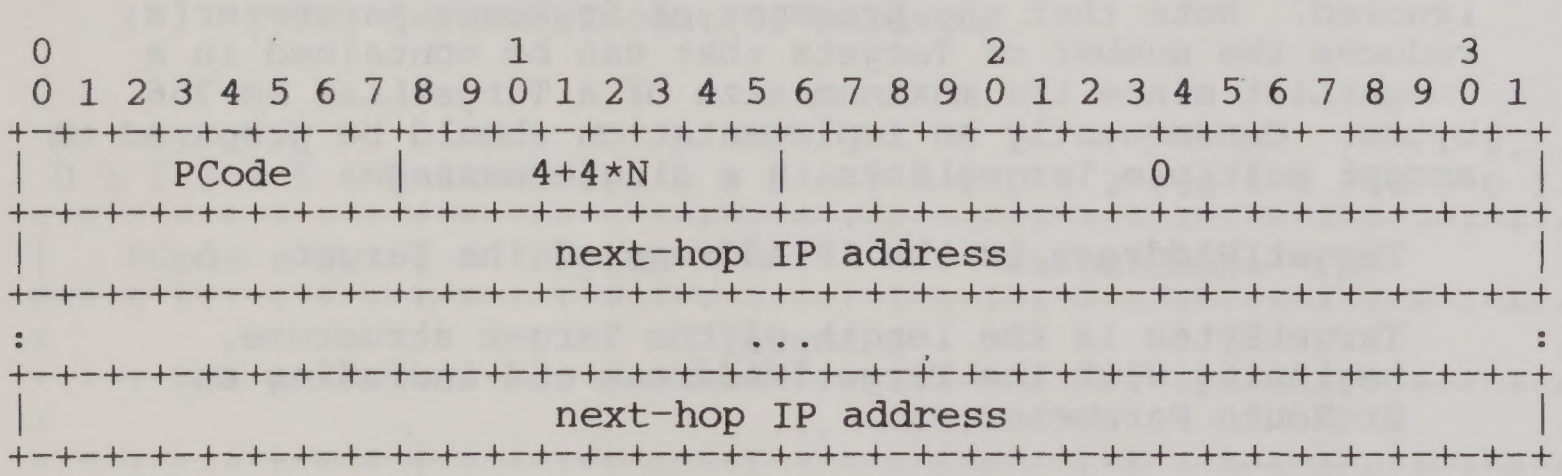


Figure 35. SrcRoute

Since it is possible that a single hop between ST agents is actually composed of multiple IP hops using IP encapsulation, it might be necessary to also specify an IP source routing option. Two additional PCodes are used in this case. See [15] for a description of IP routing options.

An IP Loose Source Route (PCode = 16) indicates that PDUs for the next-hop ST agent should be encapsulated in IP and that the IP datagram should contain an IP Loose Source Route constructed from the list of IP router addresses contained in this option.

An IP Strict Source Route (PCode = 17) is similarly used when the corresponding IP Strict Source Route option should be constructed.

Consequently, the "routing parameter" may consist of a sequence of one or more separate parameters with PCodes 16, 17, 18, or 19.

4.2.2.15. Target and TargetList

Several control messages use a parameter called TargetList (PCode = 20), which contains information about the targets to which the message pertains. For each Target in the TargetList, the information includes the IP addresses of the target, the SAP applicable to the next higher layer protocol, the length of the SAP (SAPBytes), and zero or more optional SrcRoute parameters; see Section 4.2.2.14 (page 95). Consequently, a Target structure can be of variable length. Each entry has the format shown in Figure 36.

The optional SrcRoute parameter is only meaningful in a CONNECT messages; if present in other messages, they are ignored. Note that the presence of SrcRoute parameter(s) reduces the number of Targets that can be contained in a TargetList since the maximum size of a TargetList is 256 bytes. Consequently an implementation should be prepared to accept multiple TargetLists in a single message.

TargetIPAddress is the IP Address of the Target.

TargetBytes is the length of the Target structure, beginning with the TargetIPAddress and including any SrcRoute Parameter(s).

SAPBytes is the length of the SAP, excluding any padding required to maintain 32-bit alignment. I.e.,

there would be no padding required for SAPs with lengths of 2, 6, etc., bytes.

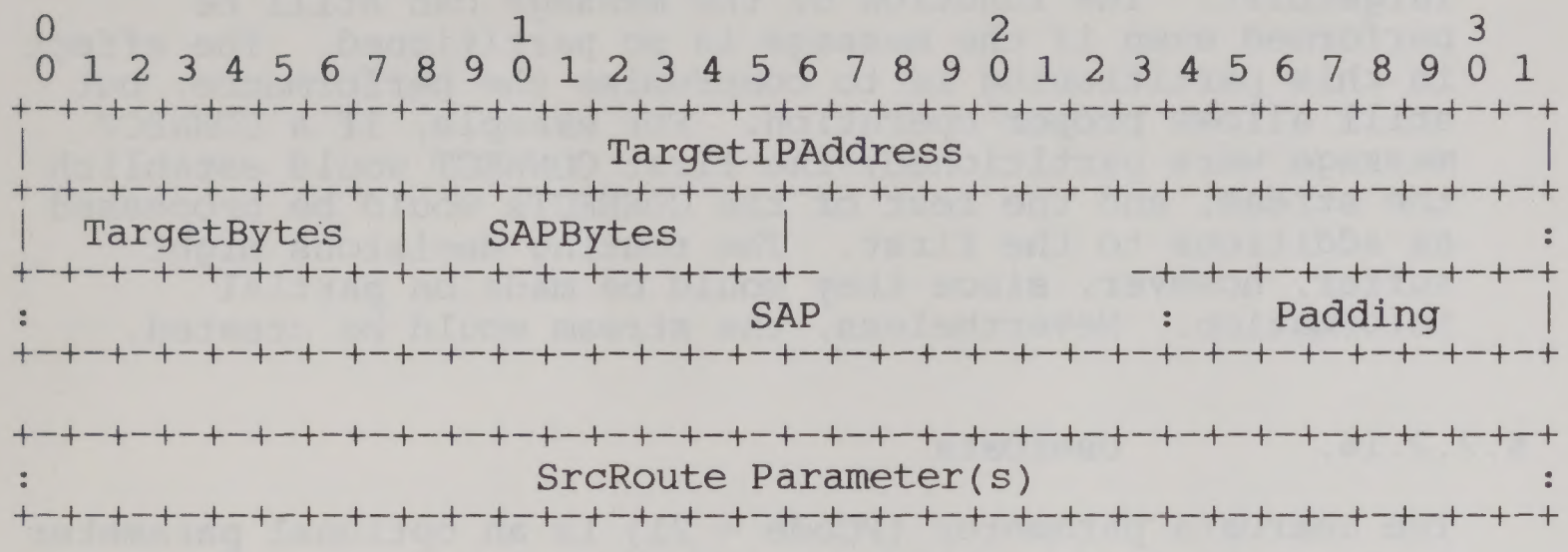


Figure 36. Target

We assume that the ST agents must know the maximum packet size of the networks to which they are connected (the MTU), and those maximum sizes will restrict the number of targets that can be specified in control messages. We feel that this is not a serious drawback. High bandwidth networks such as the Ethernet or the Terrestrial Wideband network support packet sizes large enough to allow well over one hundred targets to be specified, and we feel that conferences with a larger number of participants will not occur for quite some time. Furthermore, we expect that future higher bandwidth networks will allow even larger packet sizes. It may be desirable to send ST voice data packets in individual B-ISDN ATM cells, which are small, but network services on ATM will provide "adaptation layers" to implement network-level fragmentation that may be used to carry larger ST control messages.

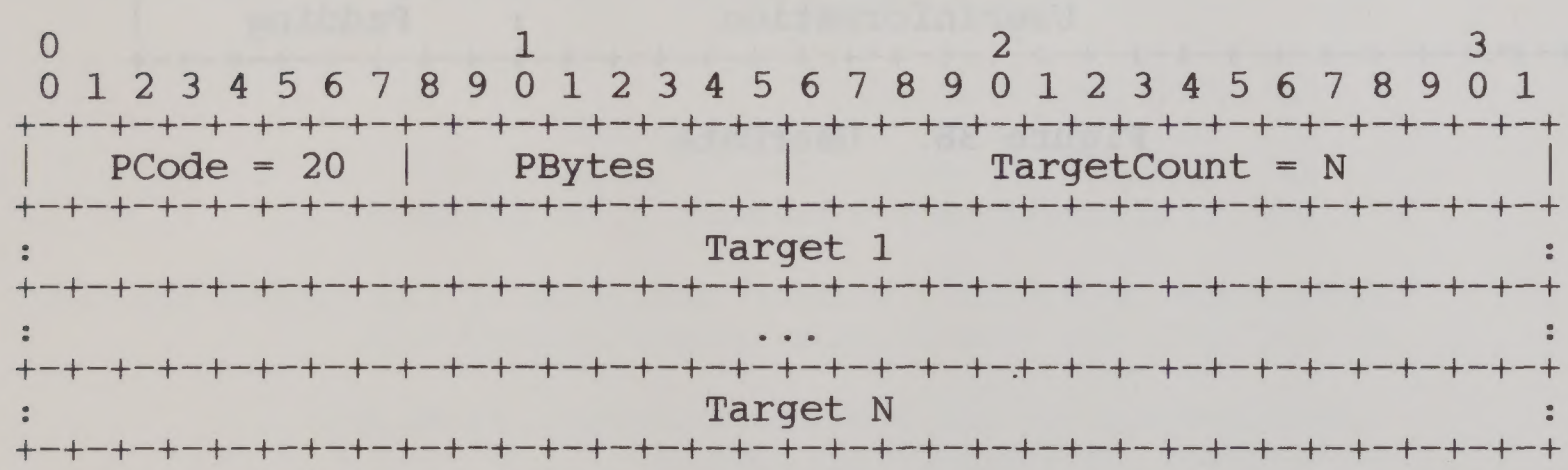


Figure 37. TargetList

If a message must pass across a network whose maximum packet size is too small, the message must be broken up into multiple messages, each of which carries part of the TargetList. The function of the message can still be performed even if the message is so partitioned. The effect in this partitioning is to compromise the performance, but still allows proper operation. For example, if a CONNECT message were partitioned, the first CONNECT would establish the stream, and the rest of the CONNECTs would be processed as additions to the first. The routing decisions might suffer, however, since they would be made on partial information. Nevertheless, the stream would be created.

4.2.2.16. UserData

The UserData parameter (PCode = 21) is an optional parameter that may be used by the next higher protocol or an application to convey arbitrary information to its peers. Note that since the size of control messages is limited by the smallest MTU in the path to the target(s), the maximum size of this parameter cannot be specified a priori. If the parameter is too large for some network's MTU, a UserDataSize error will occur. The parameter must be padded to a multiple of 32 bits.

UserBytes specifies the number of valid UserInformation bytes.

UserInformation is arbitrary data meaningful to the next higher protocol layer or application.

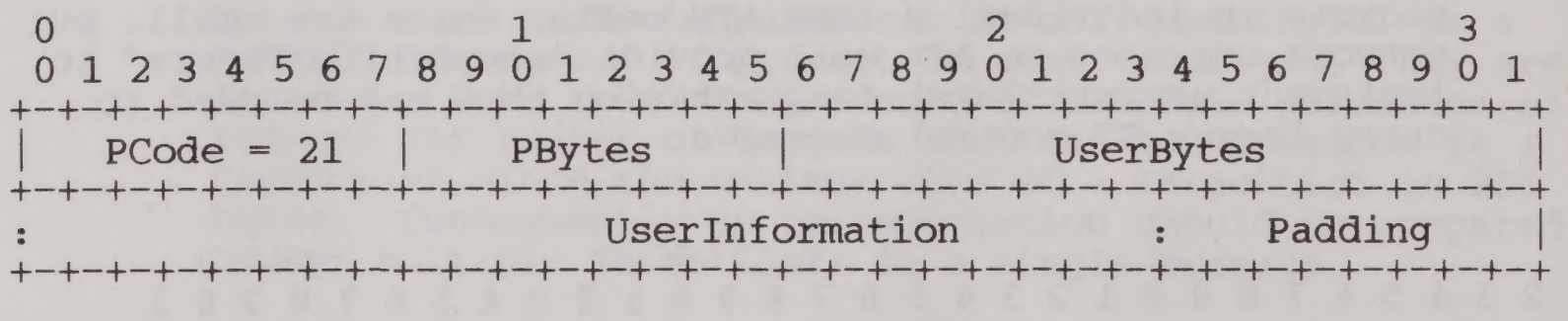


Figure 38. UserData

4.2.3. ST Control Message PDUs

Each control message is described in a following section. See Appendix 1 (page 147) for an explanation of the notation.

4.2.3.1. ACCEPT

ACCEPT (OpCode = 1) is issued by a target as a positive response to a CONNECT message. It implies that the target is prepared to accept data from the origin along the stream that was established by the CONNECT. The ACCEPT includes the FlowSpec that contains the cumulative information that was calculated by the intervening ST agents as the CONNECT made its way from the origin to the target, as well as any modifications made by the application at the target. The ACCEPT is relayed by the ST agents from the target to the origin along the path established by the CONNECT but in the reverse direction. The ACCEPT must be acknowledged with an ACK at each hop.

The FlowSpec is not modified on this trip from the target back to the origin. Since the cumulative FlowSpec information can be different for different targets, no attempt is made to combine the ACCEPTs from the various targets. The TargetList included in each ACCEPT contains the IP address of only the target that issued the ACCEPT.

Any SrcRoute parameters in the TargetList are ignored.

Since an ACCEPT might be the first response from a next-hop on a control link (due to network reordering), the SVLId field may be the first source of the Virtual Link Identifier to be used in the RVLId field of subsequent control messages sent to that next-hop.

When the FDx option has been selected to setup a second stream in the reverse direction, the ACCEPT will contain both RFlowSpec and RName parameters. Each agent should update the state tables for the reverse stream with this information.

TSR (bits 14 and 15) specifies the target's response for the use of data packet timestamps; see Section 4 (page 76). Its values and semantics are:

00 Not implemented.
01 No timestamps are permitted.
10 Timestamps must always be present.
11 Timestamps may optionally be present.

Reference contains a number assigned by the agent sending the ACCEPT for use in the acknowledging ACK.

LnkReference is the Reference number from the corresponding CONNECT or CHANGE.

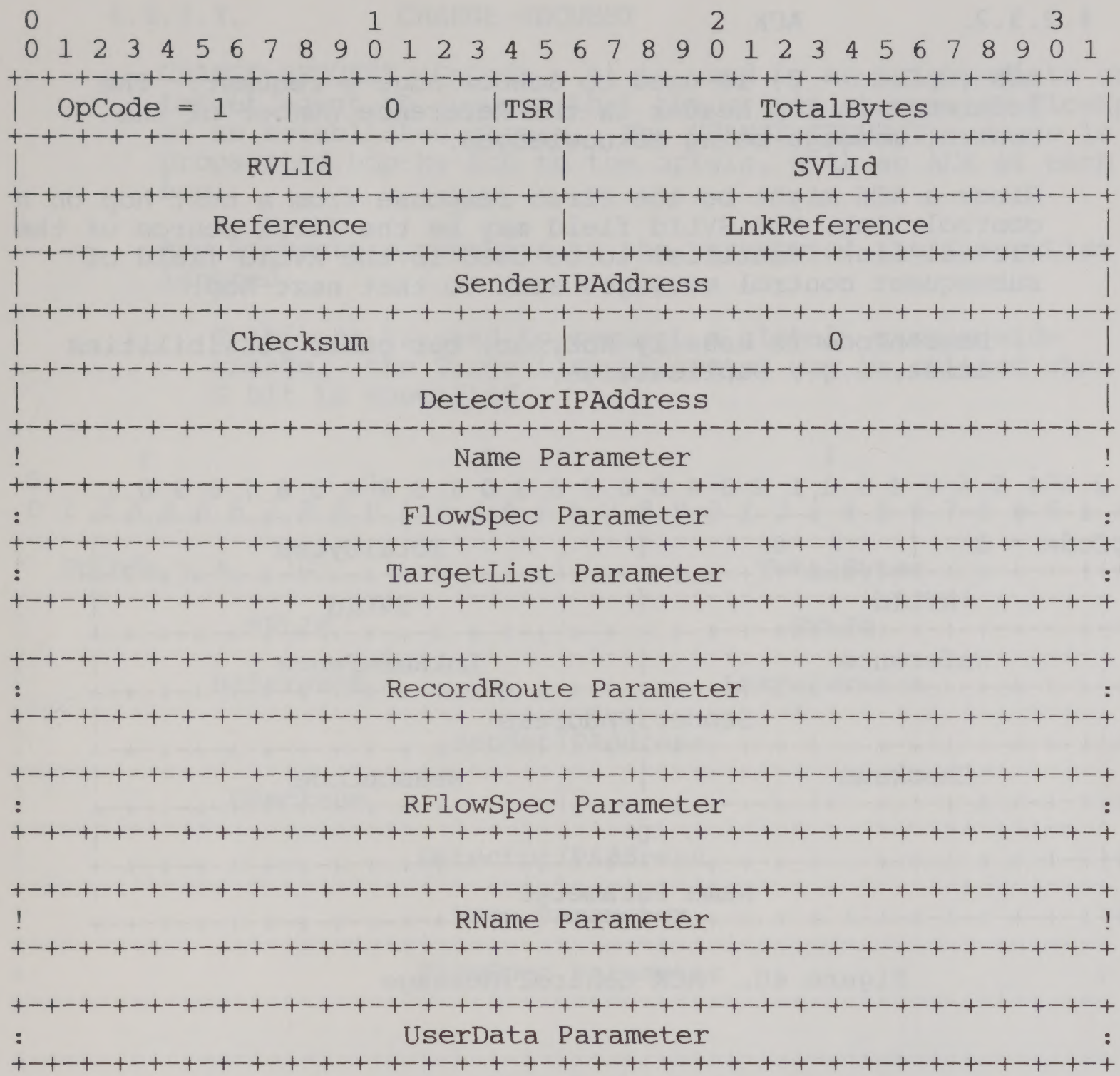


Figure 39. ACCEPT Control Message

4.2.3.2. ACK

ACK (OpCode = 2) is used to acknowledge a request. The Reference in the header is the Reference number of the control message being acknowledged.

Since a ACK might be the first response from a next-hop on a control link, the SVLId field may be the first source of the Virtual Link Identifier to be used in the RVLId field of subsequent control messages sent to that next-hop.

ReasonCode is usually NoError, but other possibilities exist, e.g., DuplicateIgn.

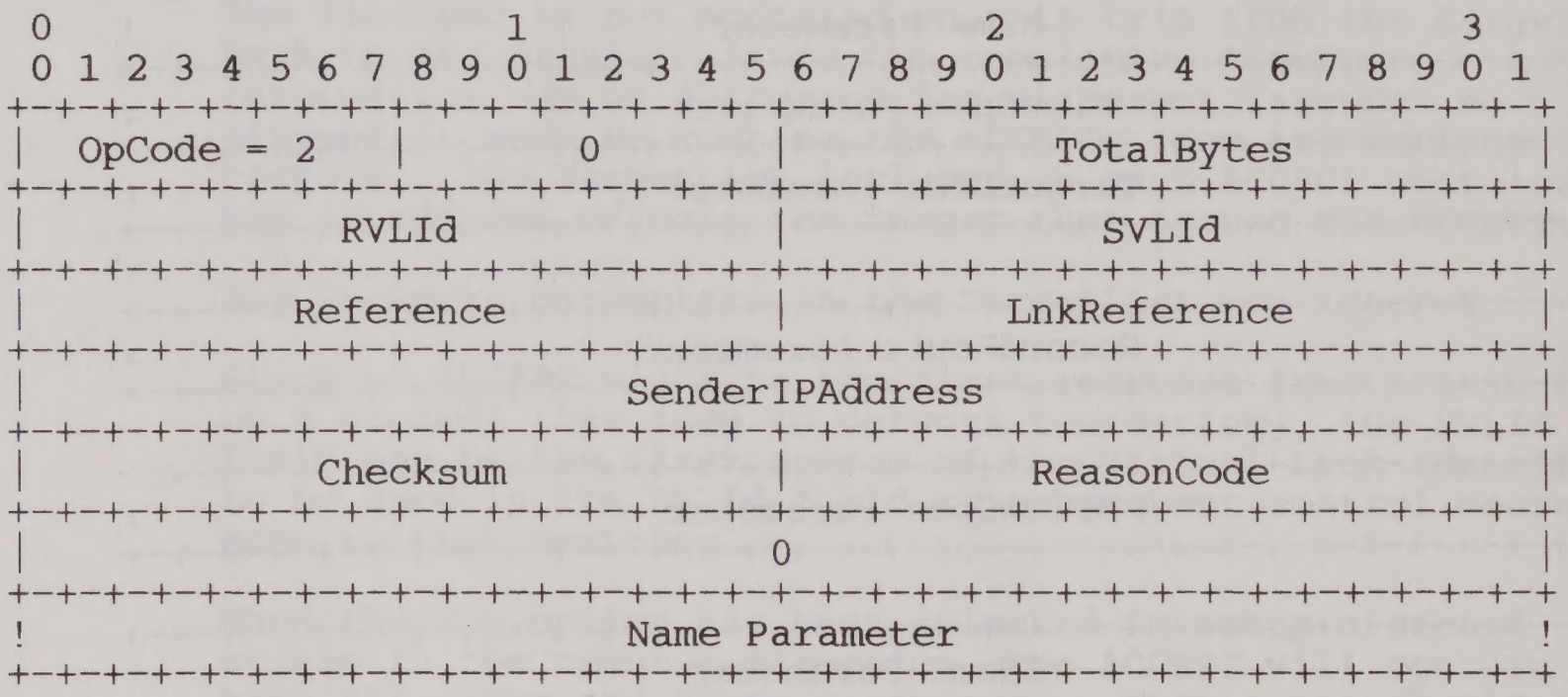


Figure 40. ACK Control Message

4.2.3.3. CHANGE-REQUEST

CHANGE-REQUEST (OpCode = 4) is used by an intermediate or target agent to request that the origin change the FlowSpec of an established stream. The CHANGE-REQUEST message is propagated hop-by-hop to the origin, with an ACK at each hop.

Any SrcRoute parameters in the targets of the TargetList are ignored.

G (bit 8) is used to request a global, stream-wide change; the TargetList parameter may be omitted when the G bit is specified.

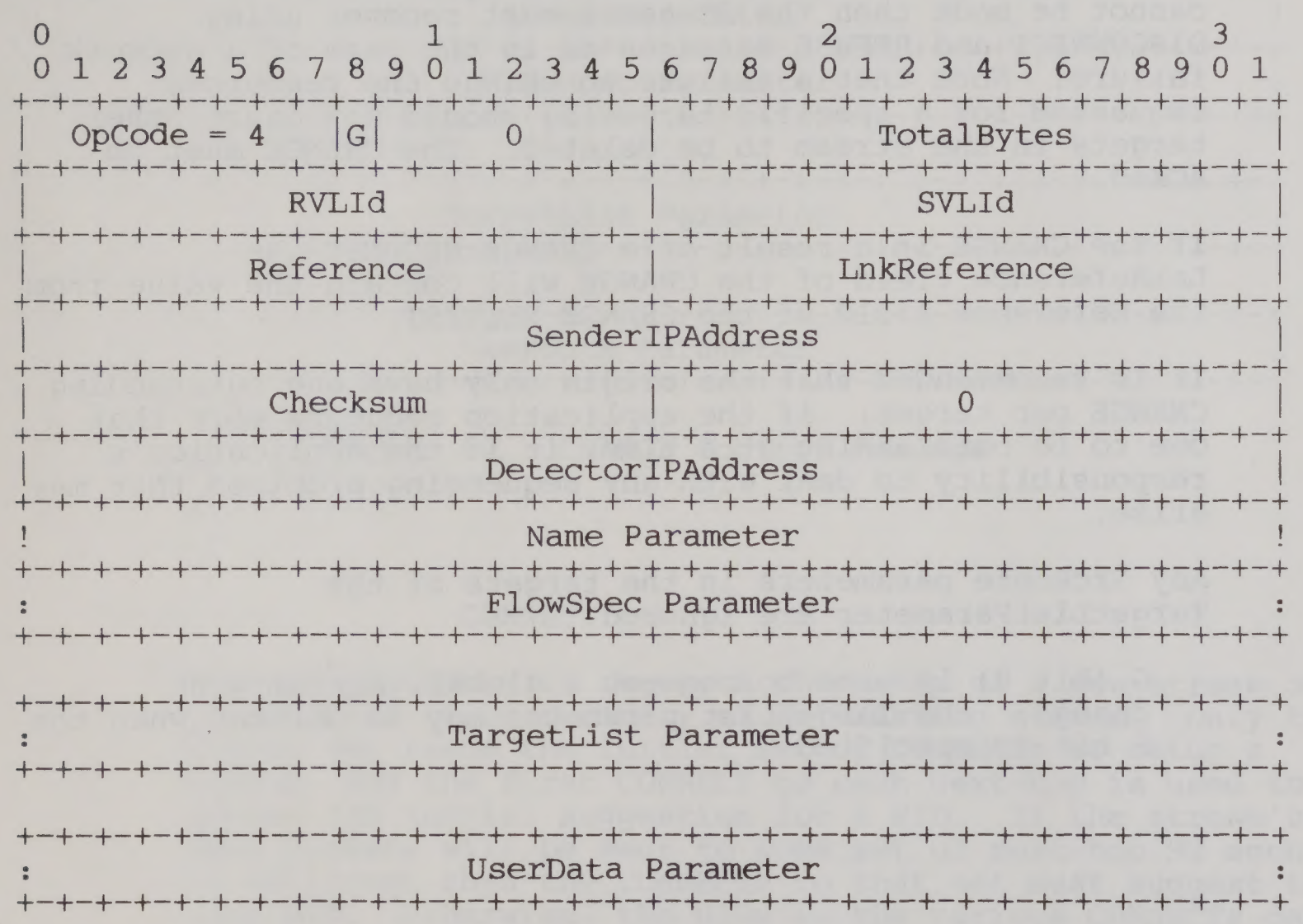


Figure 41. CHANGE-REQUEST Control Message

4.2.3.4. CHANGE

CHANGE (OpCode = 3) is used to change the FlowSpec of an established stream. Parameters are the same as for CONNECT but the TargetList is not required. The CHANGE message is processed similarly to the CONNECT message, except that it travels along the path of an established stream.

If the change to the FlowSpec is in a direction that makes fewer demands of the involved networks, then the change has a high probability of success along the path of the established stream. Each ST agent receiving the CHANGE message makes the necessary requested changes to the network resource allocations, and if successful, propagates the CHANGE message along the established paths. If the change cannot be made then the ST agent must recover using DISCONNECT and REFUSE messages as in the case of a network failure. Note that a failure to change the resources requested for a specific target(s) should not cause other targets in the stream to be deleted. The CHANGE must be ACKed.

If the CHANGE is a result of a CHANGE-REQUEST the LnkReference field of the CHANGE will contain the value from the Reference field of the CHANGE-REQUEST.

It is recommended that the origin only have one outstanding CHANGE per target; if the application requests more that one to be outstanding at a time, it is the application's responsibility to deal with any sequencing problems that may arise.

Any SrcRoute parameters in the targets of the TargetListParameter are ignored.

G (bit 8) is used to request a global, stream-wide change; the TargetList parameter may be omitted when the G bit is specified.

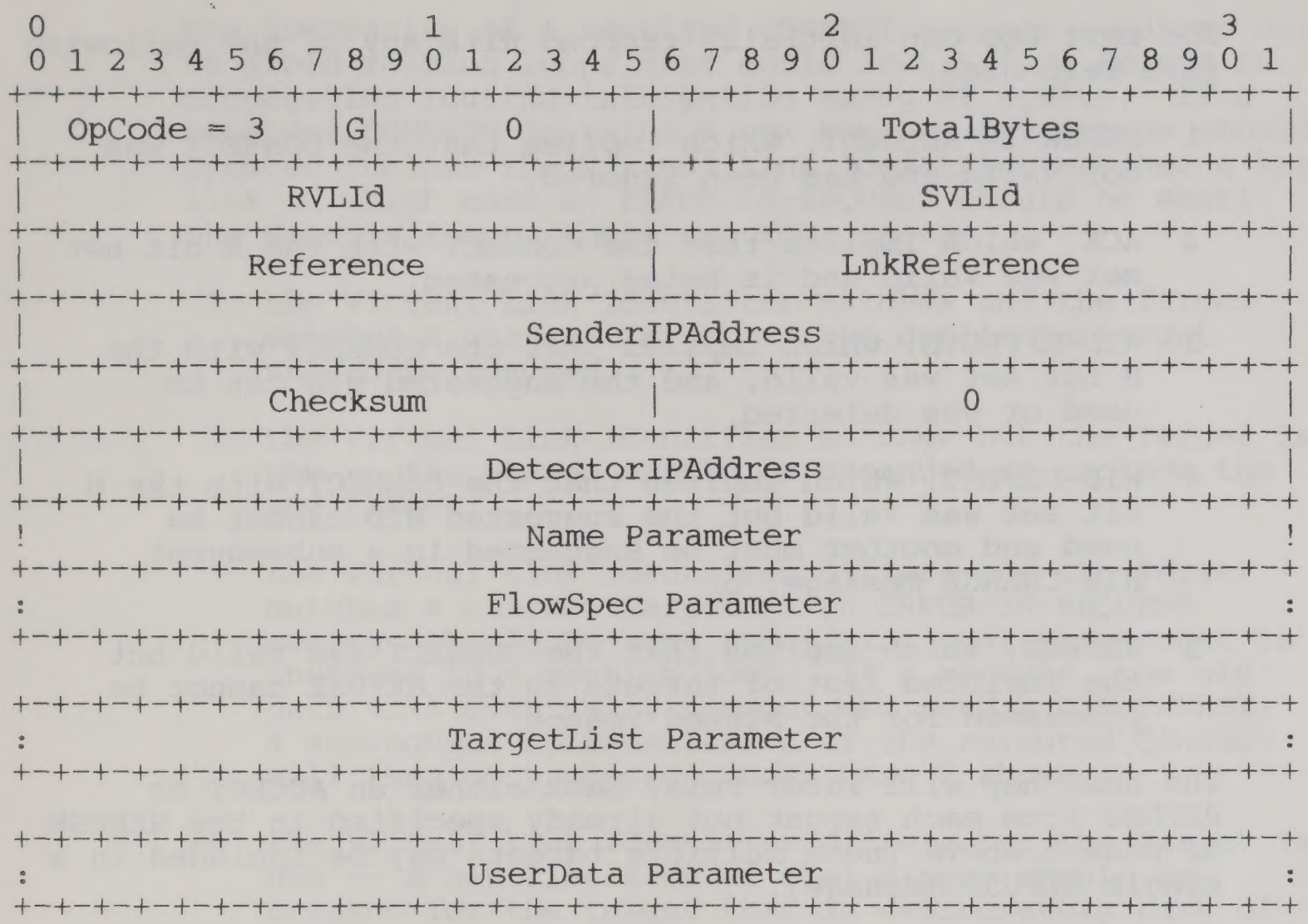


Figure 42. CHANGE Control Message

4.2.3.5. CONNECT

CONNECT (OpCode = 5) requests the setup of a new stream or an addition to or recovery of an existing stream. Only the origin can issue the initial set of CONNECTs to setup a stream, and the first CONNECT to each next-hop is used to convey the initial suggestion for a HID. If the stream's data packets will be sent to some set of next-hop ST agents by multicast then the CONNECTs to that set must suggest the same HID. Otherwise, the HIDs in the various CONNECTs can be different.

The CONNECT message must fit within the maximum allowable packet size (MTU) for the intervening network. If a CONNECT message is too large, it must be fragmented into multiple CONNECT messages by partitioning the TargetList; see Section 4.2 (page 77). Any UserData parameter will be replicated in each fragment for delivery to all targets.

The next-hop can initially respond with any of the following five responses:

1 ERROR-IN-REQUEST, which implies that the CONNECT was not valid and has been ignored,

2 ACK, which implies that the CONNECT with the H bit not set was valid and is being processed,

3 HID-APPROVE, which implies that the CONNECT with the H bit set was valid, and the suggested HID can be used or was deferred,

4 HID-REJECT, which implies that the CONNECT with the H bit set was valid but the suggested HID cannot be used and another must be suggested in a subsequent HID-CHANGE message, or

5 REFUSE, which implies that the CONNECT was valid but the included list of targets in the REFUSE cannot be processed for the stated reason.

The next-hop will later relay back either an ACCEPT or REFUSE from each target not already specified in the REFUSE of case 5 above (note multiple targets may be included in a single REFUSE message).

An intermediate ST agent that receives a CONNECT selects the next-hop ST agents, partitions the TargetList accordingly, reserves network resources in the direction toward the next-hop, updating the FlowSpec accordingly (see Section 4.2.2.3 (page 81)), selects a proposed HID for each next-hop, and sends the resulting CONNECTs.

If the intermediate ST agent that is processing a CONNECT fails to find a route to a target, then it responds with a REFUSE with the appropriate reason code. If the next-hop to a target is by way of the network from which it received the CONNECT, then it sends a NOTIFY with the appropriate reason code (RouteBack). In either case, the TargetList specifies the affected targets. The intermediate ST agent will only route to and propagate a CONNECT to the targets for which it does not issue either an ERROR-IN-REQUEST or a REFUSE.

The processing of a received CONNECT message requires care to avoid routing loops that could result from delays in propagating routing information among ST agents. If a received CONNECT contains a new Name, a new stream should be created (unless the Virtual Link Identifier matches a known link in which case an ERROR-IN-REQUEST should be sent). If the Name is known, there are four cases:

1 the Virtual Link Identifier matches and the Target matches a current Target -- the duplicate target should be ignored.

2 the Virtual Link Identifier matches but the Target is new -- the stream should be expanded to include the new target.

3 the Virtual Link Identifier differs and the Target matches a current Target -- an ERROR-IN-REQUEST message should be sent specifying that the target is involved in a routing loop. If a reroute, the old path will eventually timeout and send a DISCONNECT; a subsequent retransmission of the rerouted CONNECT will then be processed under case 2 above.

4 the Virtual Link Identifier differs but the Target is new -- a new (instance of the) stream should be created for the target that is deliberately part of a loop using a SrcRoute parameter.

Note that the test for a known or matching Target includes comparing any SrcRoute parameter that might be present.

Option bits are specified by either the origin's service user or by an intermediate agent, depending on the specific option. Bits not specified below are currently unspecified, and should be set to zero (0) by the origin agent and not changed by other agents unless those agents know their meaning.

H (bit 8) is used for the HID Field option; see Section 3.6.1 (page 44). It is set to one (1) only if the HID field contains either zero (when the HID selection is being deferred), or the proposed HID. This bit is zero (0) if the HID field does not contain valid data and should be ignored.

P (bit 9) is used for the PTP option; see Section 3.6.2 (page 44).

S (bit 10) is used for the NoRecovery option; see Section 3.6.4 (page 46).

TSP (bits 14 and 15) specifies the origin's proposal for the use of data packet timestamps; see Section 4 (page 76). Its values and semantics are:

00 No proposal.
01 Cannot insert timestamps.
10 Must always insert timestamps.
11 Can insert timestamps if requested.

RVLId, the receiver's Virtual Link Identifier, is set to zero in all CONNECT messages until its value arrives in the SVLId field of an acknowledgment to the CONNECT.

SVLId, the sender's Virtual Link Identifier, is set to a value chosen by each hop to facilitate efficient dispatching of subsequent control messages.

HID is the identifier that will be used with data packets moving through the stream in the direction from the origin to the targets. It is a hop-by-hop shorthand identifier for the stream's Name, and is chosen by each agent for the branch to the next-hop agents. The contents of the HID field are only valid, and a HID-REJECT or HID-APPROVE reply may only be sent, when the HID Field option (H bit) is set (1). If the HID Field option is specified and the proposed HID is zero, the selection of the HID is deferred to the receiving next-hop agent. If the HID Field option is not set (H bit is 0), then the HID field does not contain valid data and should be ignored; see Section 3.6.1 (page 44).

TargetList is the list of IP addresses of the target processes. It is of arbitrary size up to the maximum allowed for packets traveling across the specific network.

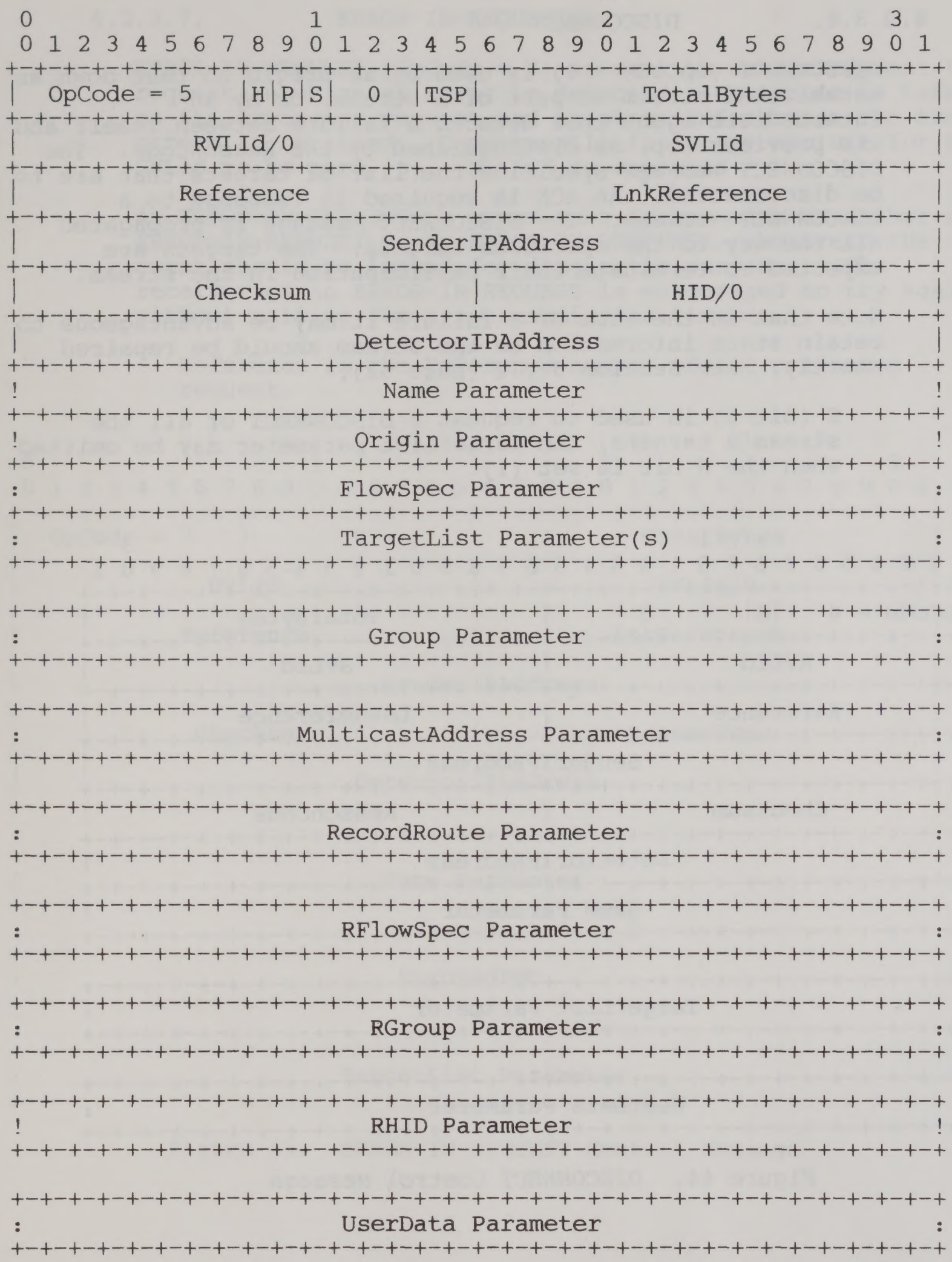


Figure 43. CONNECT Control Message

4.2.3.6. DISCONNECT

DISCONNECT (OpCode = 6) is used by an origin to tear down an established stream or part of a stream, or by an intermediate agent that detects a failure between itself and its previous-hop, as distinguished by the ReasonCode. The DISCONNECT message specifies the list of targets that are to be disconnected. An ACK is required in response to a DISCONNECT message. The DISCONNECT message is propagated all the way to the specified targets. The targets are expected to terminate their participation in the stream.

Note that in the case of a failure it may be advantageous to retain state information as the stream should be repaired shortly; see Section 3.7.2 (page 52).

G (bit 8) is used to request a DISCONNECT of all the stream's targets; the TargetList parameter may be omitted when the G bit is set (1).

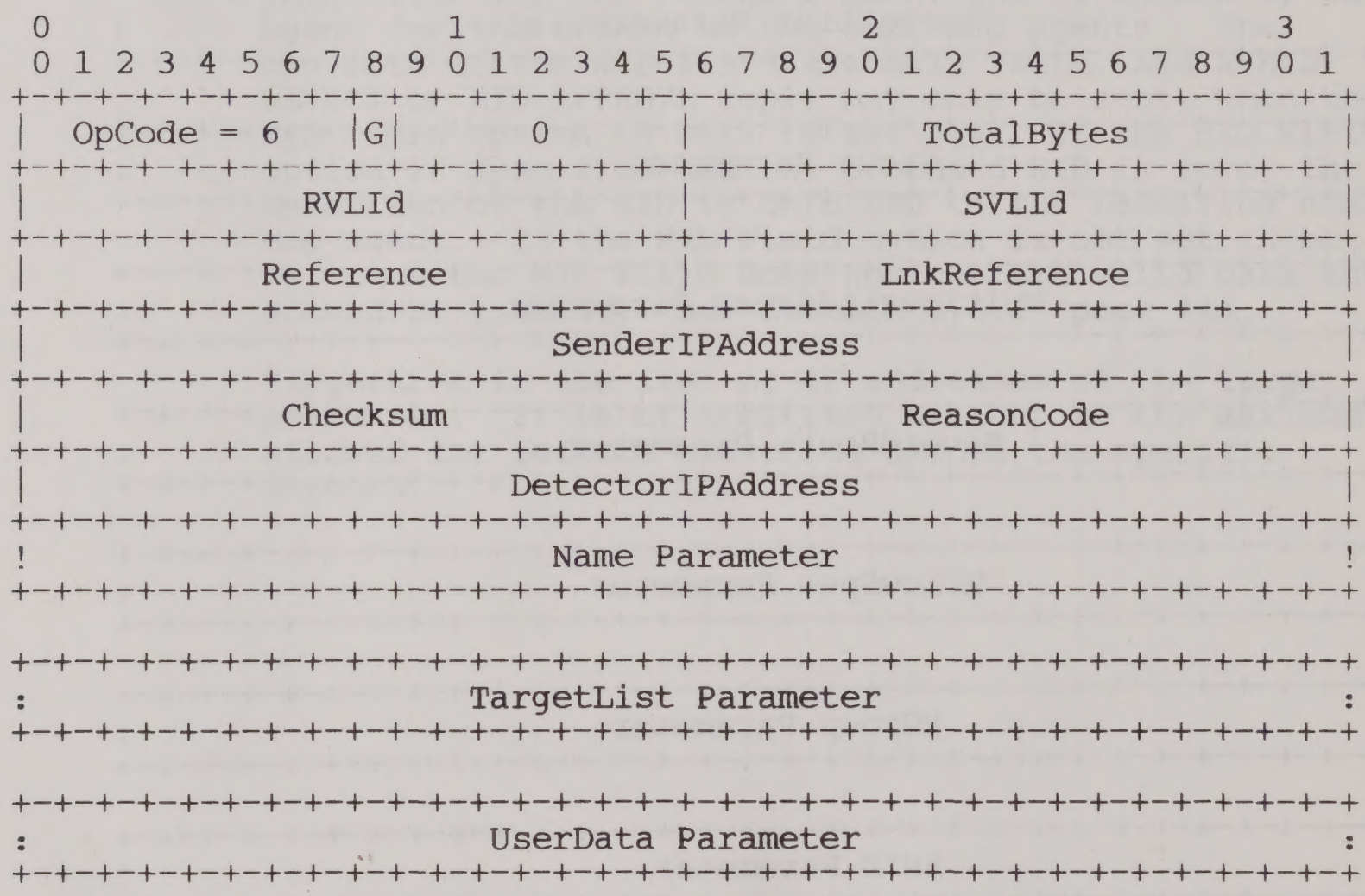


Figure 44. DISCONNECT Control Message

4.2.3.7. ERROR-IN-REQUEST

ERROR-IN-REQUEST (OpCode = 7) is sent in acknowledgment to a request in which an error is detected. No action is taken on the erroneous request and no state information for the stream is retained. Consequently it is appropriate for the SVLId to be zero (0). No ACK is expected.

An ERROR-IN-REQUEST is never sent in response to either an ERROR-IN-REQUEST or an ERROR-IN-RESPONSE; however, the event should be logged for diagnostic purposes. The receiver of an ERROR-IN-REQUEST is encouraged to try again without waiting for a retransmission timeout.

Reference is the Reference number of the erroneous request.

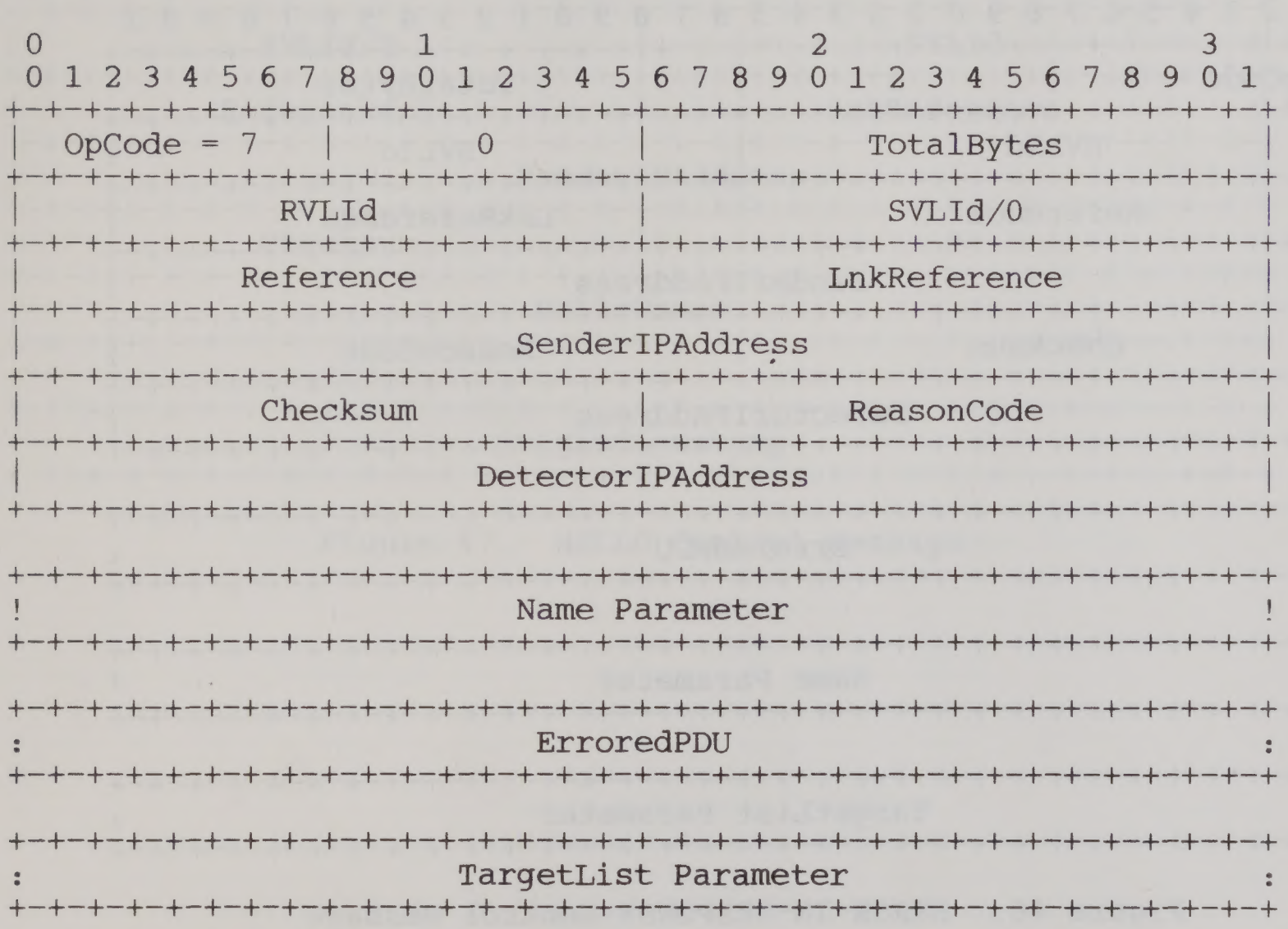


Figure 45. ERROR-IN-REQUEST Control Message

4.2.3.8. ERROR-IN-RESPONSE

ERROR-IN-RESPONSE (OpCode = 8) is sent in acknowledgment to a response in which an error is detected. No ACK is expected. Action taken by the requester and responder will vary with the nature of the request.

An ERROR-IN-REQUEST is never sent in response to either an ERROR-IN-REQUEST or an ERROR-IN-RESPONSE; however, the event should be logged for diagnostic purposes. The receiver of an ERROR-IN-RESPONSE is encouraged to try again without waiting for a retransmission timeout.

Reference identifies the erroneous response.

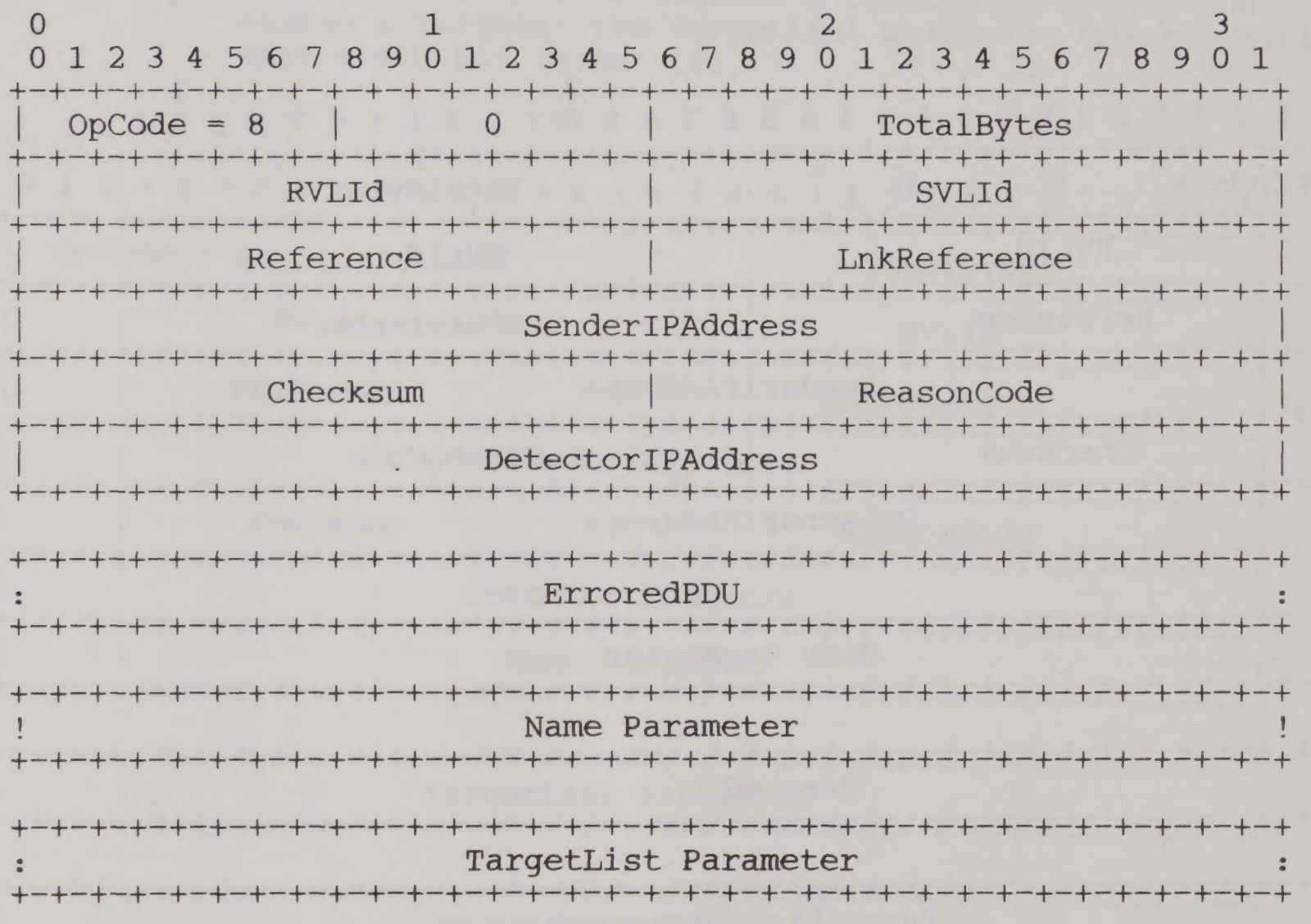


Figure 46. ERROR-IN-RESPONSE Control Message

4.2.3.9. HELLO

HELLO (OpCode = 9) is used as part of the ST failure detection mechanism; see Section 3.7.1.2 (page 49).

R (bit 8) is used for the Restarted bit.

Reference is non-zero to inform the receiver that an ACK should be promptly sent so that the sender can update its round-trip time estimates. If the Reference is zero, no ACK should be sent.

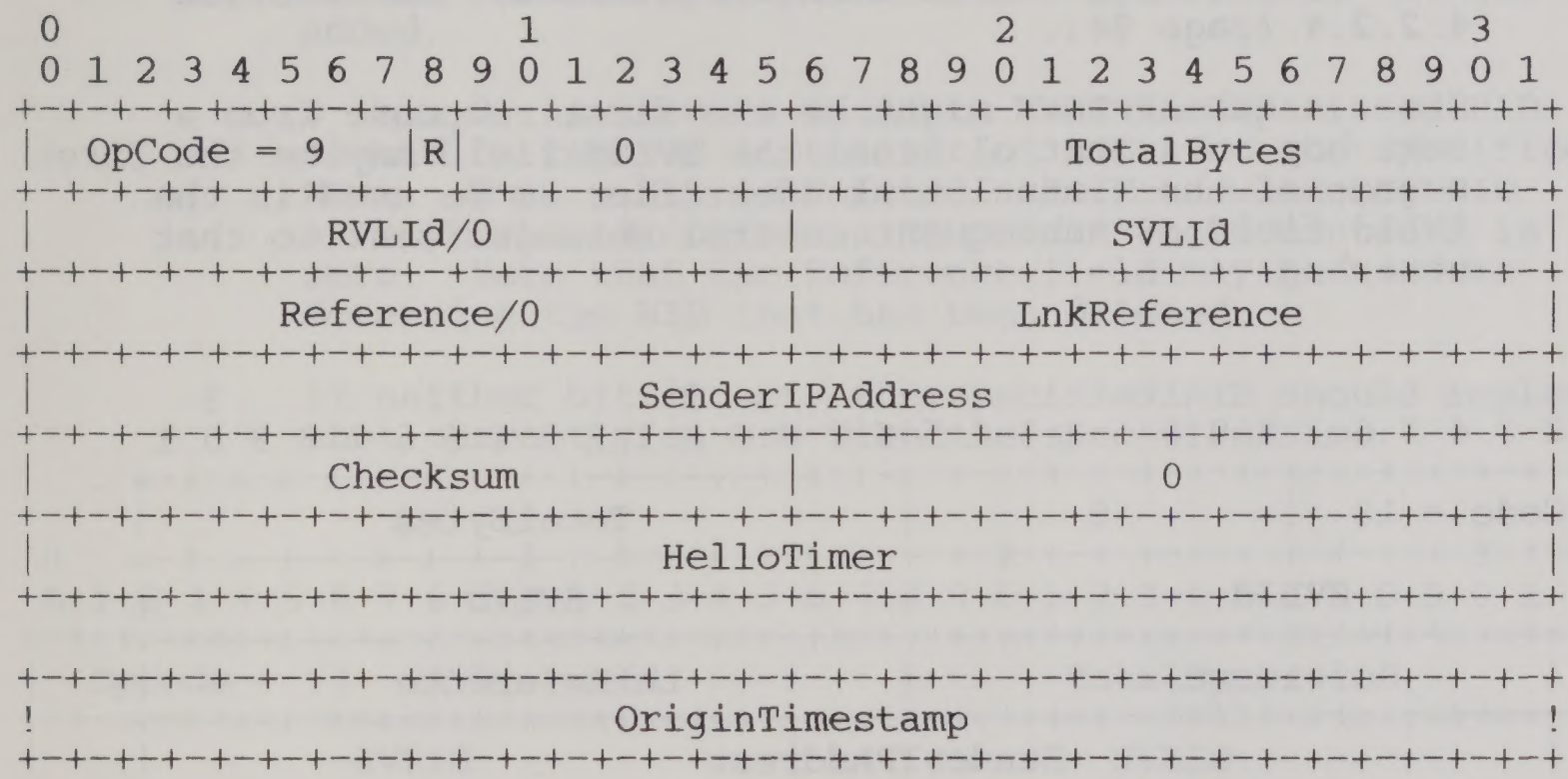


Figure 47. HELLO Control Message

4.2.3.10. HID-APPROVE

HID-APPROVE (OpCode = 10) is used by the agent that is responding to either a CONNECT or HID-CHANGE to agree to either use the proposed HID or to the addition or deletion of the specified HID. In all cases but deletion, the newly approved HID is returned in the HID field; for deletion, the HID field must be set to zero. The HID-APPROVE is the acknowledgment of a CONNECT or HID-CHANGE.

The optional FreeHIDs parameter provides the previous-hop agent with hints about what other HIDs are acceptable in case a multicast HID is being negotiated; see Section 4.2.2.4 (page 84).

Since a HID-APPROVE might be the first response from a next-hop on a control link, the SVLId field may be the first source of the Virtual Link Identifier to be used in the RVLId field of subsequent control messages sent to that next-hop.

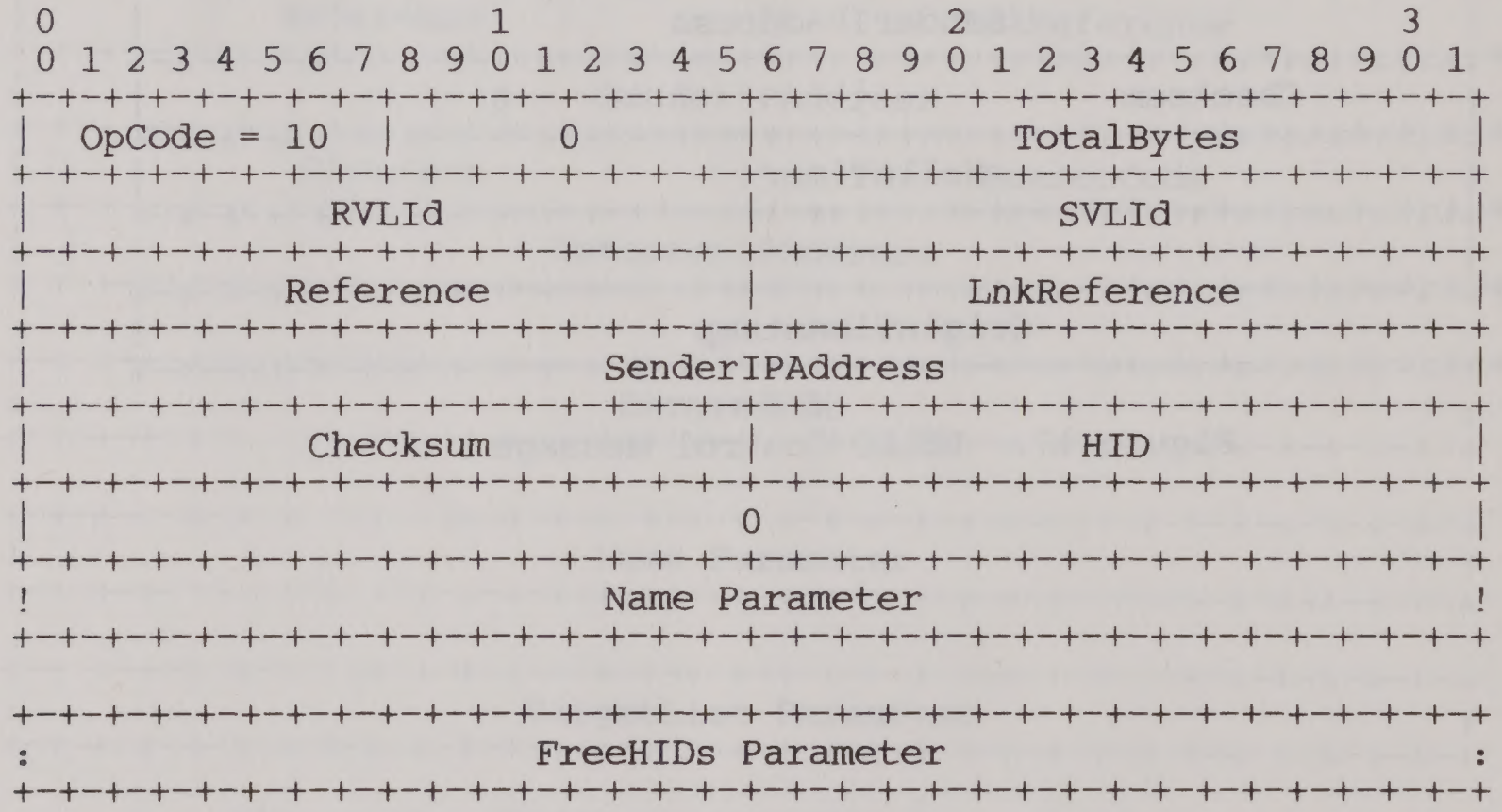


Figure 48. HID-APPROVE Control Message

4.2.3.11. HID-CHANGE-REQUEST

HID-CHANGE-REQUEST (OpCode = 12) is used by a next-hop agent that would like, for administrative reasons, to change the HID that is in use. The receiving previous-hop agent acknowledges the request by either an ERROR-IN-REQUEST if it is unwilling to make the requested change, or with a HID-CHANGE if it can accommodate the request.

A (bit 8) is used to indicate that the specified HID should be included in the set of HIDs for the specified Name. When a HID is added, the acknowledging HID-APPROVE should contain a HID field whose contents is the HID just added.

D (bit 9) is used to indicate that the specified HID should be removed in the set of HIDs for the specified Name. When a HID is deleted, the acknowledging HID-APPROVE should contain a HID field whose contents is zero. Note that the Reference field may be used to determine the HID that has been deleted.

If neither bit is set, the specified HID should replace that currently in use with the specified Name.

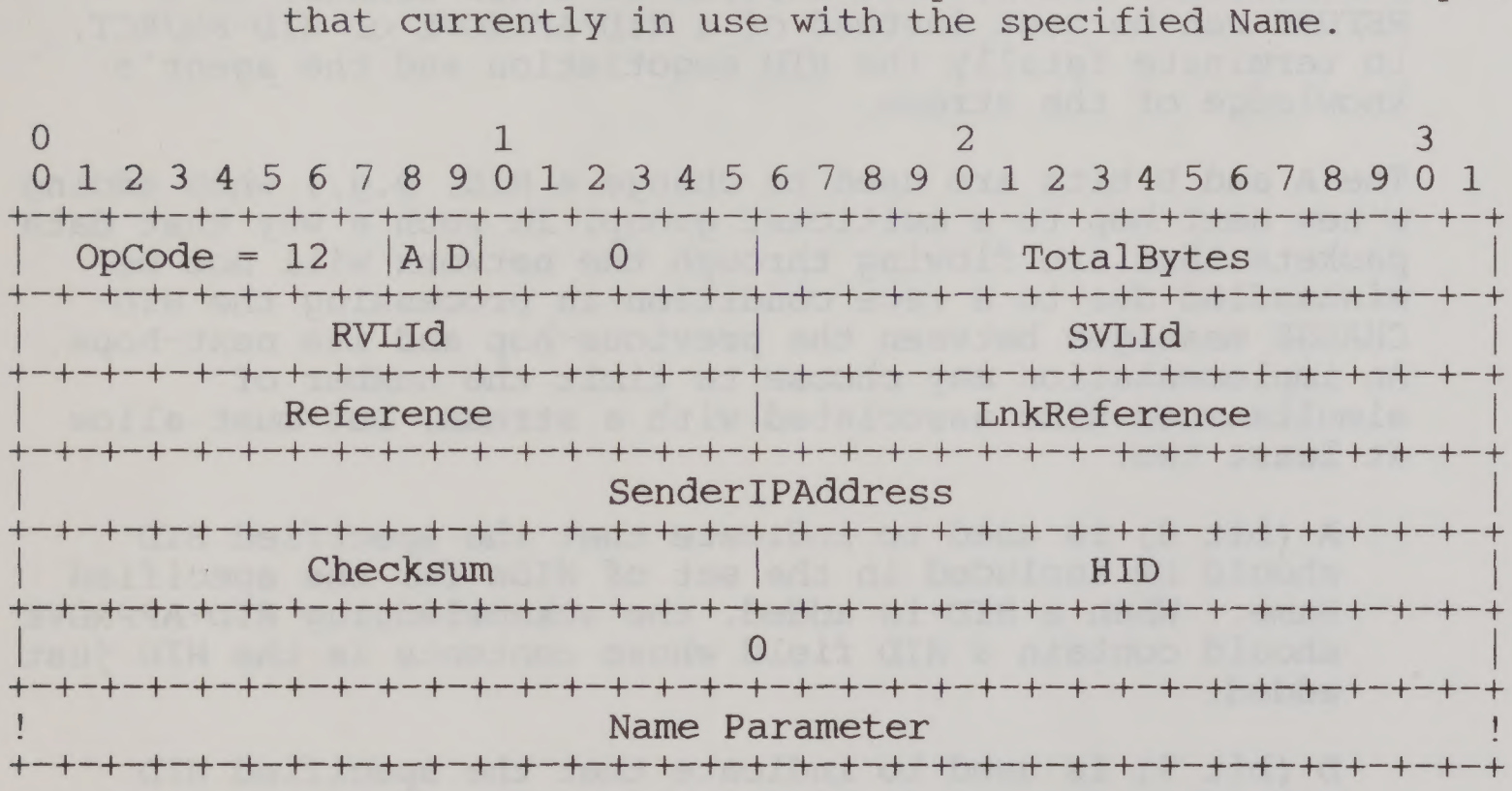


Figure 49. HID-CHANGE-REQUEST Control Message

4.2.3.12. HID-CHANGE

HID-CHANGE (OpCode = 11) is used by the agent that issued a CONNECT and received a HID-REJECT to attempt to negotiate a suitable HID. The HID in the HID-CHANGE message must be different from that in the CONNECT, or any previous HID-CHANGE messages for the given Name. The agent receiving the HID-CHANGE must respond with a HID-APPROVE if the new HID is suitable, or a HID-REJECT if it is not. In case of an error, either an ERROR-IN-REQUEST or a REFUSE may be returned as an acknowledgment.

Since an agent may send CONNECT messages with the same HID to several next-hops in order to use multicast data transfer, any HID-CHANGE must also be sent to the same set of next-hops. Therefore, a next-hop agent must be prepared to receive a HID-CHANGE before or after it has sent a HID-APPROVE response to the CONNECT or a previous HID-CHANGE. Only the last HID-CHANGE is relevant. The previous-hop agent will ignore HID-APPROVE or HID-REJECT messages to previous CONNECT or HID-CHANGE messages.

A DISCONNECT can be sent instead of a HID-CHANGE, or a REFUSE can be sent instead of a HID-APPROVE or HID-REJECT, to terminate fatally the HID negotiation and the agent's knowledge of the stream.

The A and D bits are used to change a HID, e.g., when adding a new next-hop to a multicast group, in such a way that data packets that are flowing through the network will not be mishandled due to a race condition in processing the HID-CHANGE messages between the previous-hop and its next-hops. An implementation may choose to limit the number of simultaneous HIDs associated with a stream, but must allow at least two.

A (bit 8) is used to indicate that the specified HID should be included in the set of HIDs for the specified Name. When a HID is added, the acknowledging HID-APPROVE should contain a HID field whose contents is the HID just added.

D (bit 9) is used to indicate that the specified HID should be removed from the set of HIDs for the specified Name. When a HID is deleted, the acknowledging HID-APPROVE should contain a HID field whose contents is zero. Note that the Reference field may be used to determine the HID that has been deleted.

If neither bit is set, the specified HID should replace that currently in use for the specified Name.

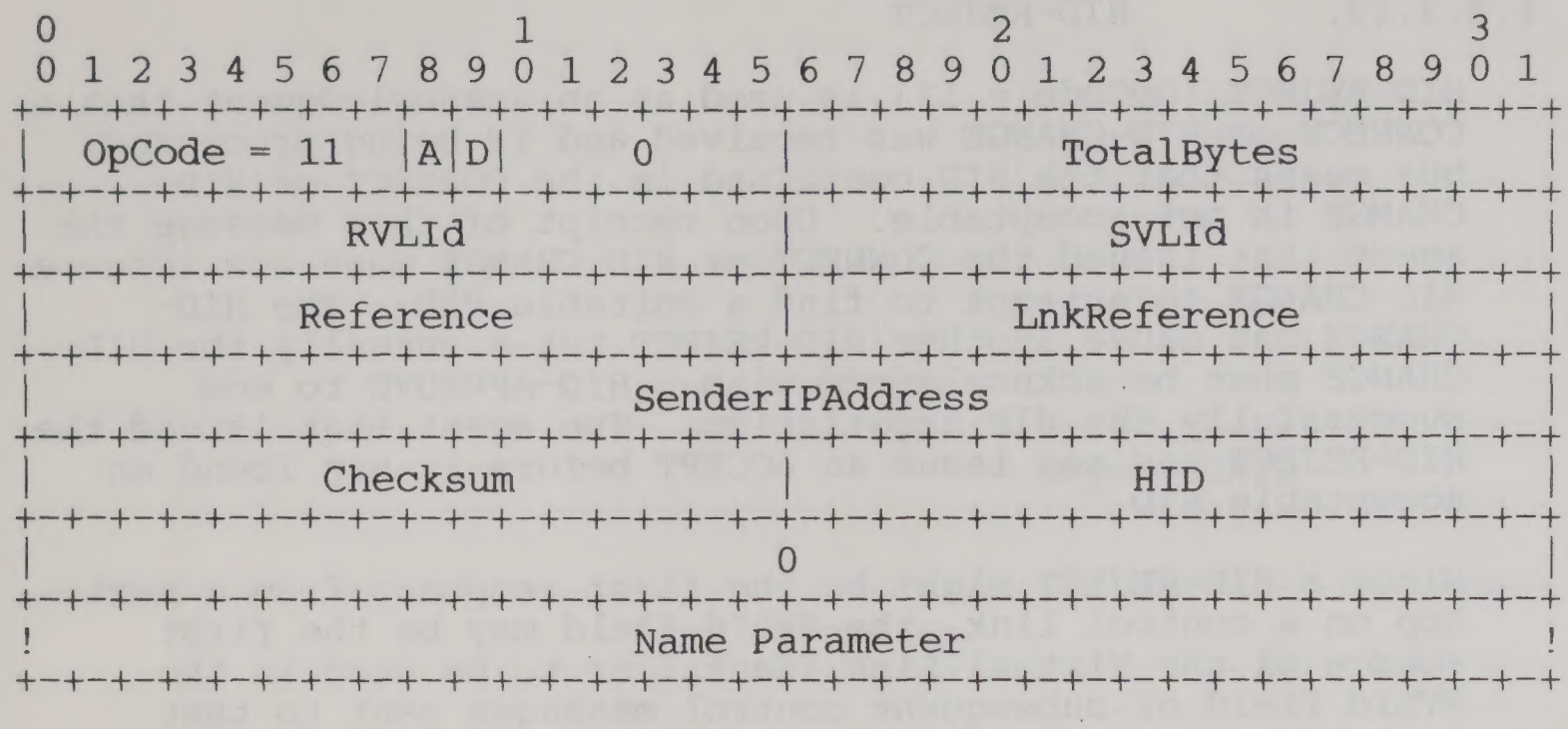


Figure 50. HID-CHANGE Control Message

4.2.3.13. HID-REJECT

HID-REJECT (OpCode = 13) is used as an acknowledgment that a CONNECT or HID-CHANGE was received and is being processed, but means that the HID contained in the CONNECT or HID-CHANGE is not acceptable. Upon receipt of this message the agent that issued the CONNECT or HID-CHANGE must now issue a HID-CHANGE to attempt to find a suitable HID. The HID-CHANGE can cause another HID-REJECT but eventually the HID-CHANGE must be acknowledged with a HID-APPROVE to end successfully the HID negotiation. The agent that issued the HID-REJECT may not issue an ACCEPT before it has found an acceptable HID.

Since a HID-REJECT might be the first response from a next-hop on a control link, the SVLId field may be the first source of the Virtual Link Identifier to be used in the RVLId field of subsequent control messages sent to that next-hop.

Either agent may terminate the negotiation by issuing either a DISCONNECT or a REROUTE. The agent that issued the HID-REJECT may issue a REFUSE, or REROUTE at any time after the HID-REJECT. In this case, the stream cannot be created, the HID negotiation need not proceed, and the previous-hop need not transmit any further messages; any further messages that are received should be ignored.

The optional FreeHIDs parameter provides the previous-hop agent with hints about what HIDs would have been acceptable; see Section 4.2.2.4 (page 84).

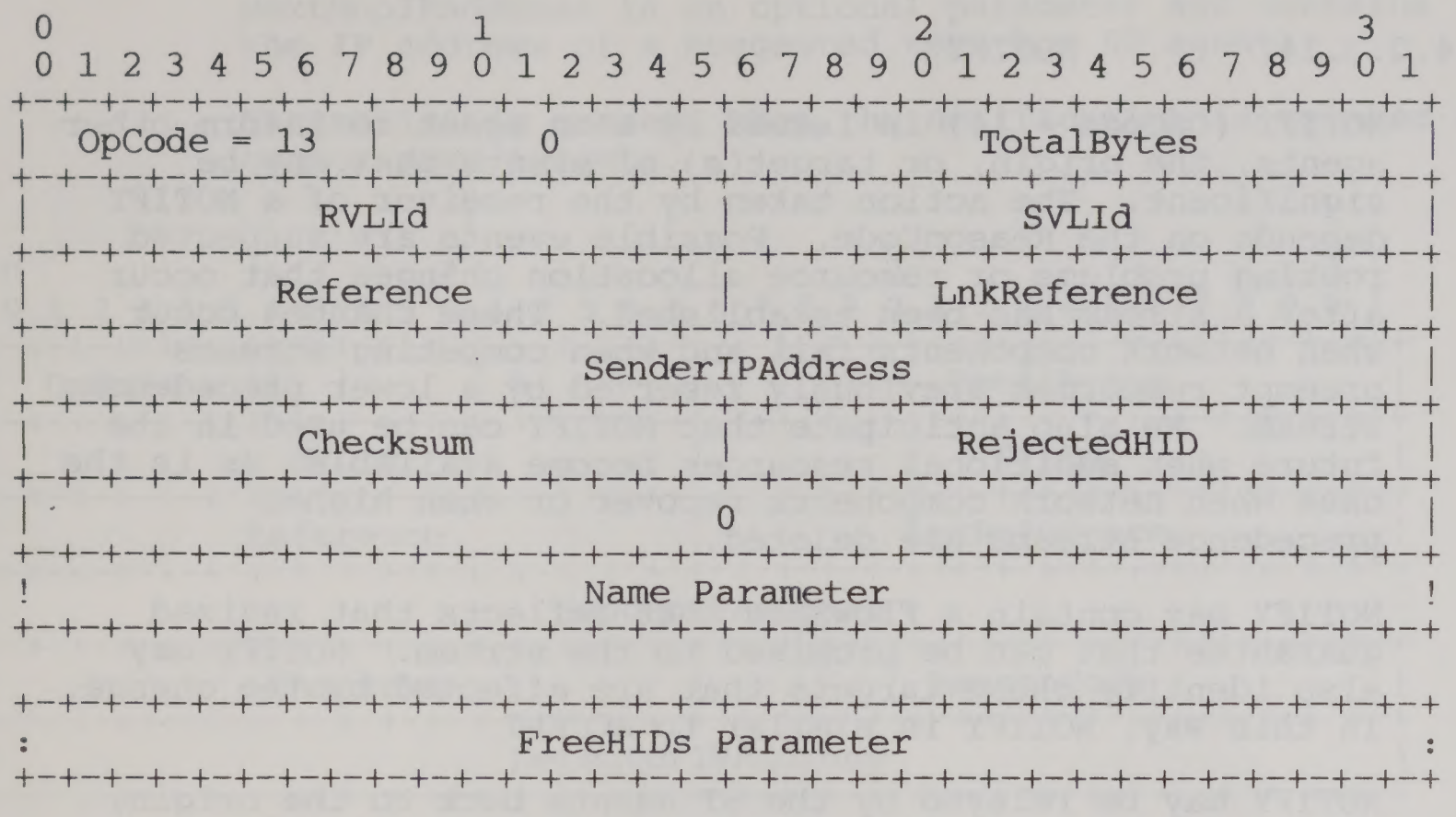


Figure 51. HID-REJECT Control Message

4.2.3.14. NOTIFY

NOTIFY (OpCode = 14) is issued by a an agent to inform other agents, the origin, or target(s) of events that may be significant. The action taken by the receiver of a NOTIFY depends on the ReasonCode. Possible events are suspected routing problems or resource allocation changes that occur after a stream has been established. These changes occur when network components fail and when competing streams preempt resources previously reserved by a lower precedence stream. We also anticipate that NOTIFY can be used in the future when additional resources become available, as is the case when network components recover or when higher precedence streams are deleted.

NOTIFY may contain a FlowSpec that reflects that revised guarantee that can be promised to the stream. NOTIFY may also identify those targets that are affected by the change. In this way, NOTIFY is similar to ACCEPT.

NOTIFY may be relayed by the ST agents back to the origin, along the path established by the CONNECT but in the reverse direction. It is up to the origin to decide whether a CHANGE should be submitted.

When NOTIFY is received at the origin, the application should be notified of the target and the change in resources allocated along the path to it, as specified in the FlowSpec contained in the NOTIFY message. The application may then use the information to either adjust or terminate the portion of the stream to each affected target.

The NOTIFY may be propagated beyond the previous-hop or next-hop agent; it must be acknowledged with an ACK.

Reference contains a number assigned by the agent sending the NOTIFY for use in the acknowledging ACK.

ReasonCode identifies the reason for the notification.

LnkReference, when non-zero, is the Reference number from a command that is the subject of the notification.

HID is present when the notification is related to a HID.

Name is present when the notification is related to a stream.

NextHopIPAddress is an optional parameter and contains the IP address of a suggested next-hop ST agent.

TargetList is present when the notification is related to one or more targets.

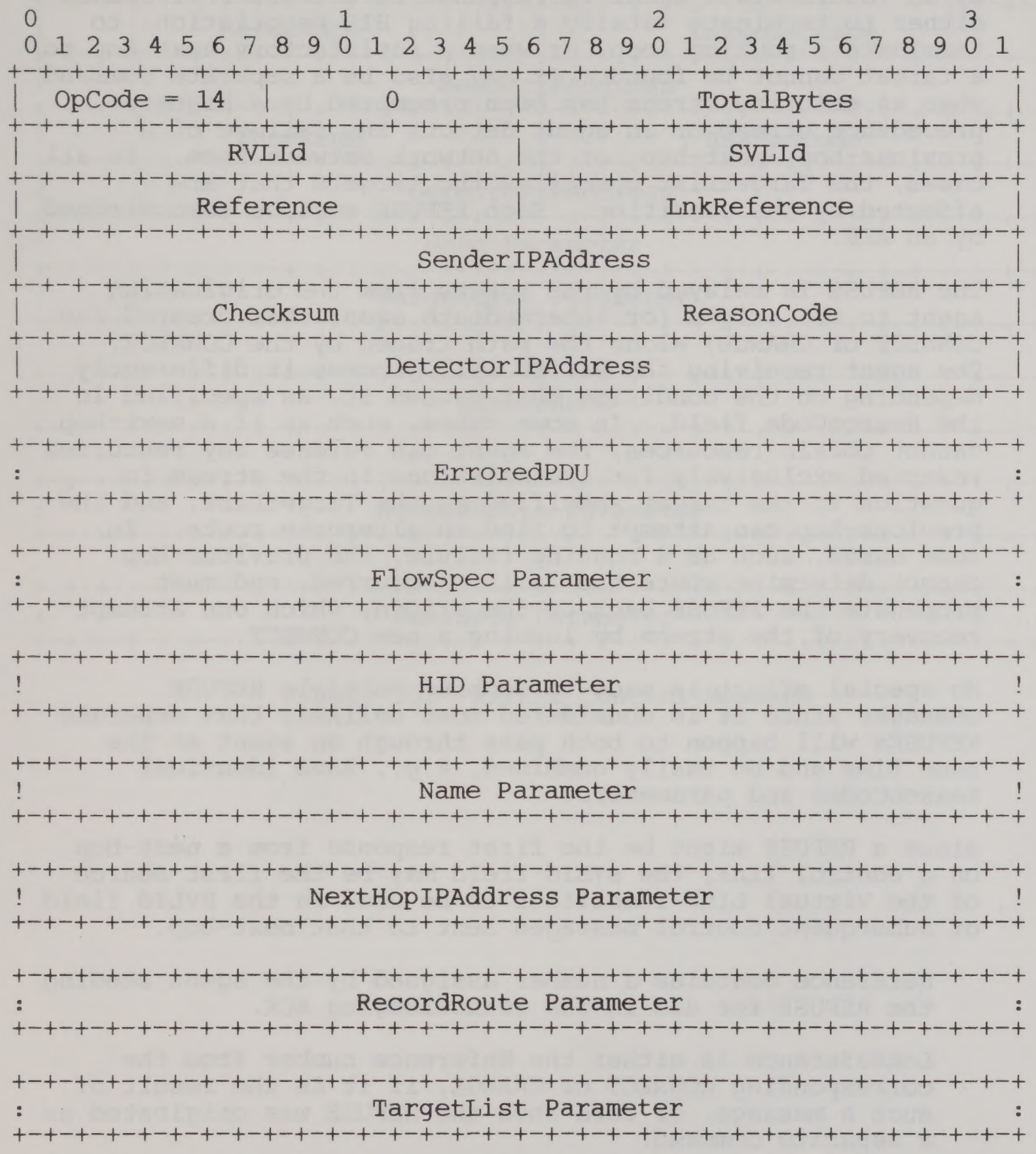


Figure 52. NOTIFY Control Message

4.2.3.15. REFUSE

REFUSE (OpCode = 15) is issued by a target that either does not wish to accept a CONNECT message or wishes to remove itself from an established stream. It might also be issued by an intermediate agent in response to a CONNECT or CHANGE either to terminate fatally a failing HID negotiation, to terminate a routing loop, or when a satisfactory next-hop to a target cannot be found. It may also be a separate command when an existing stream has been preempted by a higher precedence stream or an agent detects the failure of a previous-hop, next-hop, or the network between them. In all cases, the TargetList specifies the targets that are affected by the condition. Each REFUSE must be acknowledged by an ACK.

The REFUSE is relayed by the agents from the originating agent to the origin (or intermediate agent that created the CONNECT or CHANGE) along the path traced by the CONNECT. The agent receiving the REFUSE will process it differently depending on the condition that caused it, as specified in the ReasonCode field. In some cases, such as if a next-hop cannot obtain resources, the agent can release any resources reserved exclusively for transmissions in the stream in question to the target specified in the TargetList, and the previous-hop can attempt to find an alternate route. In some cases, such as a routing failure, the previous-hop cannot determine where the failure occurred, and must propagate the REFUSE back to the origin, which can attempt recovery of the stream by issuing a new CONNECT.

No special effort is made to combine multiple REFUSE messages since it is considered most unlikely that separate REFUSEs will happen to both pass through an agent at the same time and be easily combined, e.g., have identical ReasonCodes and parameters.

Since a REFUSE might be the first response from a next-hop on a control link, the SVLId field may be the first source of the Virtual Link Identifier to be used in the RVLId field of subsequent control messages sent to that next-hop.

Reference contains a number assigned by the agent sending the REFUSE for use in the acknowledging ACK.

LnkReference is either the Reference number from the corresponding CONNECT or CHANGE, if it is the result of such a message, or zero when the REFUSE was originated as a separate command.

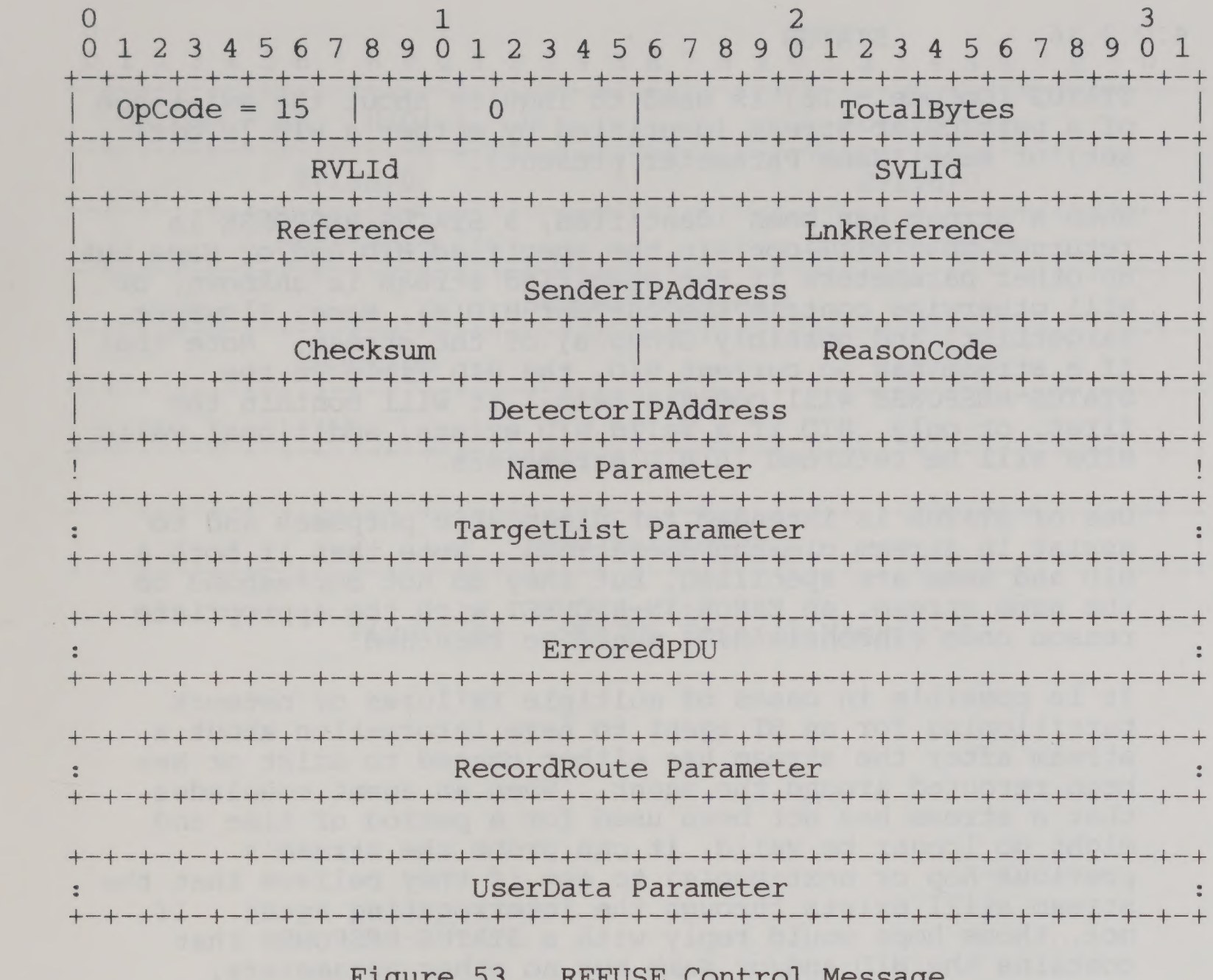


Figure 53. REFUSE Control Message

4.2.3.16. STATUS

STATUS (OpCode = 16) is used to inquire about the existence of a particular stream identified by either a HID (H bit set) or Name (Name Parameter present).

When a stream has been identified, a STATUS-RESPONSE is returned that will contain the specified HID and/or Name but no other parameters if the specified stream is unknown, or will otherwise contain the current HID(s), Name, FlowSpec, TargetList, and possibly Group(s) of the stream. Note that if a stream has no current HID, the HID field in the STATUS-RESPONSE will contain zero; it will contain the first, or only, HID if a valid HID exists; additional valid HIDs will be returned in HID parameters.

Use of STATUS is intended for diagnostic purposes and to assist in stream cleanup operations. Note that if both a HID and Name are specified, but they do not correspond to the same stream, an ERROR-IN-REQUEST with the appropriate reason code (InconsistHID) would be returned.

It is possible in cases of multiple failures or network partitioning for an ST agent to have information about a stream after the stream has either ceased to exist or has been rerouted around the agent. When an agent concludes that a stream has not been used for a period of time and might no longer be valid, it can probe the stream's previous-hop or next-hop(s) to see if they believe that the stream still exists through the interrogating agent. If not, those hops would reply with a STATUS-RESPONSE that contains the HID and/or Name but no other parameters; otherwise, if the stream is still valid, the hops would reply with the parameters of the stream.

H (bit 8) is used to indicate whether (when 1) or not (when 0) a HID is present in the HID field.

Q (bit 9) is set to one (1) for remote diagnostic purposes when the receiving agent should return a stream's parameters, whether or not the source of the message is believed to be a previous-hop or next-hop in the specified stream. Note that this use has potential for disclosure of sensitive information.

RVLId and SVLId may either or both be zero when STATUS is used for diagnostic purposes.

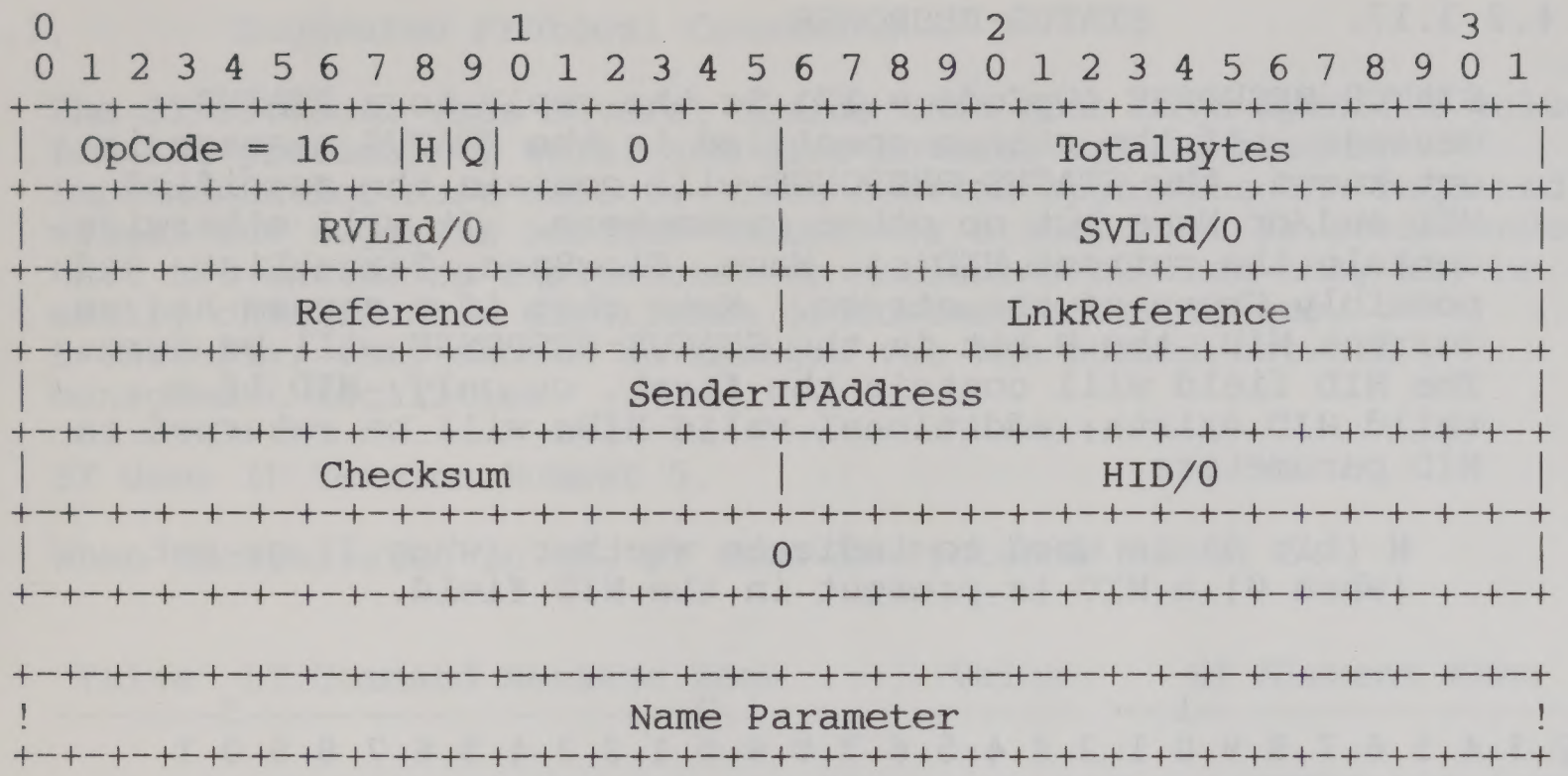


Figure 54. STATUS Control Message

4.2.3.17. STATUS-RESPONSE

STATUS-RESPONSE (OpCode = 17) is the reply to a STATUS message. If the stream specified in the STATUS message is not known, the STATUS-RESPONSE will contain the specified HID and/or Name but no other parameters. It will otherwise contain the current HID(s), Name, FlowSpec, TargetList, and possibly Group of the stream. Note that if a stream has no current HID, the H bit in the STATUS-RESPONSE will be zero. The HID field will contain the first, or only, HID if a valid HID exists; additional valid HIDs will be returned in HID parameters.

H (bit 8) is used to indicate whether (when 1) or not (when 0) a HID is present in the HID field.

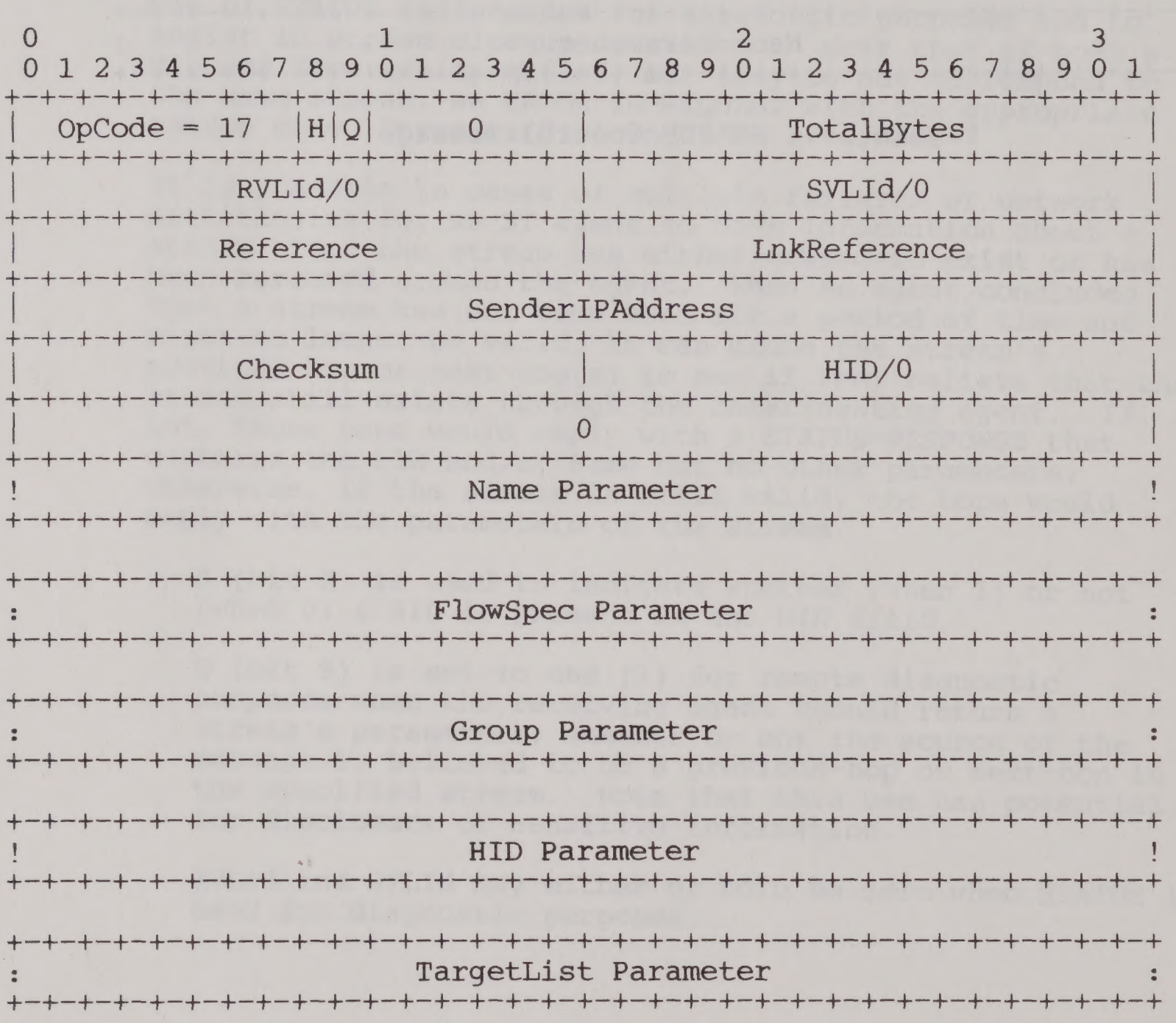


Figure 55. STATUS-RESPONSE Control Message

4.3. Suggested Protocol Constants

The ST Protocol uses several fields that must have specific values for the protocol to work, and also several values that an implementation must select. This section specifies the required values and suggests initial values for others. It is recommended that the latter be implemented as variables so that they may be easily changed when experience indicates better values. Eventually, they should be managed via the normal network management facilities.

ST uses IP Version Number 5.

When encapsulated in IP, ST uses IP Protocol Number 5.

Value	ST Command Message Name
1	ACCEPT
2	ACK
3	CHANGE
4	CHANGE-REQUEST
5	CONNECT
6	DISCONNECT
7	ERROR-IN-REQUEST
8	ERROR-IN-RESPONSE
9	HELLO
10	HID-APPROVE
11	HID-CHANGE
12	HID-CHANGE-REQUEST
13	HID-REJECT
14	NOTIFY
15	REFUSE
16	STATUS
17	STATUS-RESPONSE

Value	ST Element Name
1	ErroredPDU
2	FlowSpec
3	FreeHIDs
4	Group
5	HID
6	MulticastAddress
7	Name
8	NextHopIPAddress
9	Origin
10	OriginTimestamp
11	RecordRoute
12	RFlowSpec
13	RGroup
14	RHID
15	RName
16	SrcRoute, IP Loose
17	SrcRoute, IP Strict
18	SrcRoute, ST Loose
19	SrcRoute, ST Strict
20	TargetList
21	UserData

A good choice for the minimum number of bits in the FreeHIDBitMask element of the FreeHIDs parameter is not yet known. We suggest a minimum of 64 bits, i.e., N in Figure 25 has a value of two (2).

HID value zero (0) is reserved for ST Control Messages. HID values 1-3 are reserved for future use.

VLId value zero (0) may only be used in the RVLId field of an ST Control Message when the appropriate value has not yet been received from the other end of the virtual link;' except for an ERROR-IN-REQUEST or diagnostic message, the SVLId field may never contain a value of zero except in a diagnostic message. VLId value 1 is reserved for use with HELLO messages by those agents whose implementation wishes to have all HELLOs so identified. VLId values 2-3 are reserved for future use.

The following permanent IP multicast addresses have been assigned to ST:

224.0.0.7	All ST routers
224.0.0.8	All ST hosts

In addition, a block of transient IP multicast addresses, 224.1.0.0 - 224.1.255.255, has been allocated for ST multicast groups. Note that in the case of Ethernet, an ST Multicast address of 224.1.cc.dd maps to an Ethernet Multicast address of 01:00:5E:01:cc:dd (see [6]).

SCMP uses retransmission to effect reliability and thus has several "retransmission timers". Each "timer" is modeled by an initial time interval (ToXxx), which gets updated dynamically through measurement of control traffic, and a number of times (NXxx) to retransmit a message before declaring a failure. All time intervals are in units of milliseconds.

Value	Timeout Name	Meaning
1000	ToAccept	Initial hop-by-hop timeout for acknowledgment of ACCEPT
3	NAccept	ACCEPT retries before failure
1000	ToConnect	Initial hop-by-hop timeout for acknowledgment of CONNECT
5	NConnect	CONNECT retries before failure
1000	ToDisconnect	Initial hop-by-hop timeout for acknowledgment of DISCONNECT
3	NDisconnect	DISCONNECT retries before failure

Value	Timeout Name	Meaning
1000	ToHIDAck	Initial hop-by-hop timeout for acknowledgment of HID-CHANGE-REQUEST
3	NHIDAck	HID-CHANGE-REQUEST retries before failure
1000	ToHIDChange	Initial hop-by-hop timeout for acknowledgment of HID-CHANGE
3	NHIDChange	HID-CHANGE retries before failure
1000	ToNotify	Initial hop-by-hop timeout for acknowledgment of NOTIFY
3	NNotify	NOTIFY retries before failure
1000	ToRefuse	Initial hop-by-hop timeout for acknowledgment of REFUSE
3	NRefuse	REFUSE retries before failure
1000	ToReroute	Timeout for receipt of ACCEPT or REFUSE from targets during failure recovery
5	NReroute	CONNECT retries before failure
5000	ToEnd2End	End-to-End timeout for receipt of ACCEPT or REFUSE from targets by origin
0	NEnd2End	CONNECT retries before failure

Value	Parameter Name	Meaning
10	NHIDAbort	Number of rejected HID proposals before aborting the HID negotiation process
10000	HelloTimerHoldDown	Interval that Restarted bit must be set after ST restart
5	HelloLossFactor	Number of consecutively missed HELLO messages before declaring link failure
2000	DefaultRecoveryTimeout	Interval between successive HELLOs to/from active neighbors
2	DefaultHelloFactor	HELLO filtering function factor

5. Areas Not Addressed

There are a number of issues that will need to be addressed in the long run but are not addressed here. Some issues are network or implementation specific. For example, the management of multicast groups depends on the interface that a network provides to the ST agent, and an UP/DOWN protocol based on ST HELLO messages depends on the details of the ST agents. Both these examples may impact the ST implementations, but we feel it is inappropriate to specify them here.

In other cases we feel that appropriate solutions are not clear at this time. The following are examples of such issues:

This document does not include a routing mechanism. We do not feel that a routing strategy based on minimizing the number of hops from the source to the destination is necessarily appropriate. An alternative strategy is to minimize the consumption of internet resources within some delay constraints. Furthermore, it would be preferable if the routing function were to provide routes that incorporated bandwidth, delay, reliability, and perhaps other characteristics, not just connectivity. This would increase the likelihood that a selected route would succeed. This requirement would probably cause the ST agents to exchange more routing information than currently implemented. We feel that further research and experimentation will be required before an appropriate routing strategy is well enough defined to be incorporated into the ST specification.

Once the bandwidth for a stream has been agreed upon, it is not sufficient to rely on the origin to transmit traffic at that rate. The internet should not rely on the origin to operate properly. Furthermore, even if the origin sources traffic at the agreed rate, the packets may become aggregated unintentionally and cause local congestion. There are several approaches to addressing this problem, such as metering the traffic in each stream as it passes through each agent. Experimentation is necessary before such a mechanism is selected.

The interface between the agent and the network is very limited. A mechanism is provided by which the ST layer can query the network to determine the likelihood that a stream can be supported. However, this facility will require practical experience before its appropriate use is defined.

The simplex tree model of a stream does not easily allow for using multiple paths to support a greater bandwidth. That is, at any given point in a stream, the entire incoming bandwidth must be transmitted to the same next-hop in order to get to some target. If the bandwidth isn't available along any single path, the stream cannot be built to that target. It may be the case that the bandwidth is not available along a single path, but if the data

flow is split along multiple paths, and so multiple next-hops, sufficient bandwidth would be available. As currently specified, the ST agent at the point where the multiple flows converge will refuse the second connection because it can only be interpreted as a routing failure. A mechanism that allows multiple paths in a stream and can protect against routing failures has not been defined.

If sufficient bandwidth is not available, both preemption and rerouting are possible. However, it is not clear when to use one or the other. As currently specified, an ST agent that cannot obtain sufficient bandwidth will attempt to preempt lower precedence streams before attempting to reroute around the bottleneck. This may lead to an undesirably high number of preemptions. It may be that a higher precedence stream can be rerouted around lower precedence streams and still meet its performance requirements, whereas the preempted lower precedence streams cannot be reconstructed and still meet their performance requirements. A simple and effective algorithm to allow a better decision has not been identified.

In case a stream cannot be completed, ST does not report to the application the nature of the trouble in any great detail. Specifically, the application cannot determine where the bottleneck is, whether the problem is permanent or transitory, or the likely time before the trouble may be resolved. The application can only attempt to build the stream at some later time hoping that the trouble has been resolved. Schemes can be envisioned by which information is relayed back to the application. However, only practical experience can evaluate the kind of trouble that is most likely encountered and the nature of information that would be most useful to the application.

A mechanism is also not defined for cases where a stream cannot be completed not because of lack of resources but because of an unexpected failure that results in an ERROR-IN-REQUEST message. An ERROR-IN-REQUEST message is returned in cases when an ST agent issues a malformed control message to a neighbor. Such an occurrence is unexpected and may be caused by a bad or incomplete ST implementation. In some cases a message, such as a NOTIFY should be sent to the origin. Such a mechanism is not defined because it is not clear what information can be extracted and what the origin should do.

No special action is taken when a target is removed from a stream. Removing a target may also remove a bottleneck either in bandwidth, packet rate or packet size, but advantage of this opportunity is not taken automatically. The application may initiate a change to the stream's characteristics, but it is not in the best position to do this because the application may not know the nature of the bottleneck. The ST layer may have the best information, but a

mechanism to do this may be very complex. As a result, this concept requires further thought.

An agent simply discards a stream's data packets if it cannot forward them. The reason may be that the packets are too large or are arriving at too high a rate. Alternative actions may include an attempt to do something with the packets, such as fragmenting them, or to notify the origin of the trouble. Corrective measures may be too complex, so it may be preferable simply to notify the origin with a NOTIFY message. However, if the incoming packet rate is causing congestion, then the NOTIFY messages themselves may cause more trouble. The nature of the communication has yet to be defined.

The FlowSpec includes a cost field, but its implementation has not been identified. The units of cost can probably be defined relatively easily. Cost of bandwidth can probably also be assigned. It is not clear how cost is assigned to other functions, such as high precedence or low delay, or how cost of the components of the stream are combined together. It is clear that the cost to provide services will become more important in the near future, but it is not clear at this time how that cost is determined.

A number of parameters of the FlowSpec are intended to be used as ranges, but some may be useful as discrete values. For example, the FlowSpec may specify that bandwidth for a stream carrying voice should be reserved in a range from 16Kbps to 64Kbps because the voice codec has a variable coding rate. However, the voice codec may be varied only among certain discrete values, such as 16Kbps, 32Kbps and 64Kbps. A stream that has 48Kbps of bandwidth is no better than one with 32Kbps. The parameters of the FlowSpec where this may be relevant should optionally specify discrete values. This is being considered.

Groups are defined as a way to associate different streams, but the nature of the association is left for further study. An example of such an association is to allow streams whose traffic is inherently not simultaneous to share the same allocated resources. This may happen for example in a conference that has an explicit floor, such that only one site can generate video or audio traffic at any given time. The grouping facility can be implemented based on this specification, but the implementation of the possible uses of groups will require new functionality to be added to the ST agents. The uses for groups and the implementation to support them will be carried out as experience is gained and the need arises.

We hope that the ST we here propose will act as a vehicle to study the use and performance of stream oriented services across packet switched networks.

[This page intentionally left blank.]

6. Glossary

appropriate reason code
: This phrase refers to one or perhaps a set of reason codes that indicate why a particular action is being taken. Typically, these result from detection of errors or anomalous conditions. It can also indicate that an application component or agent has presented invalid parameters.

DefaultRecoveryTimeout
: The DefaultRecoveryTimeout is maintained by each ST agent. It indicates the default time interval to use for sending HELLO messages.

downstream
: The direction in a stream from an origin toward its targets.

element
: The fields and parameters of the ST control messages are collectively called elements.

FlowSpec
: The Flow Specification, abbreviated "FlowSpec" is used by an application to specify required and desired characteristics of the stream. The FlowSpec specifies bandwidth, delay, and reliability parameters. Both minimal requirements and desired characteristics are included. This information is then used to guide route selection and resource allocation decisions. The desired vs. required characteristics are used to guide tradeoff decisions among competing stream requests.

group
: A set of related streams can be associated as a group. This is done by generating a Group Name and assigning it to each of the related streams. The grouping information can then be used by the ST agents in making resource management and other control decisions. For example, when preemption is necessary to establish a high precedence stream, we can exploit the group information to minimize the number of stream groups that are preempted.

Group Name
: The Group Name is used to indicate that a collection of streams are related. A Group Name is structured to ensure that it is unique across all hosts: it includes the address of the host where it was generated combined with a unique number generated by that host. A timestamp is added to ensure that the overall name is unique over all time. (A Group Name has the same format as a stream Name.)

HelloLossFactor
The HelloLossFactor is a parameter maintained by each ST agent. It identifies the expected number of consecutive HELLO messages typically lost due to transient factors. Thus, an agent will be assumed to be down after we miss more than HelloLossFactor messages.

HelloTimer
The HelloTimer is a millisecond timer maintained by each ST agent. It is included in each HELLO message. It represents the time since the agent was restarted, modulo the precision of the field. It is used to detect variations in the delay between the two agents, by comparing the arrival interval of two HELLO messages to the difference between their HelloTimer fields.

HelloTimerHoldDown
The HelloTimerHoldDown value is maintained by each ST agent. When an ST agent is restarted, it will set the "Restarted" bit in all HELLO messages it sends for HelloTimerHoldDown seconds.

HID
The Hop IDentifier, abbreviated as HID, is a numeric key stored in the header of each ST packet. It is used by an ST agent to associate the packet with one of the incoming hops managed by the agent. It can be used by receiving agent to map to the set of outgoing next-hops to which the message should be forwarded. The HID field of an ST packet will generally need to be changed as it passes through each ST agent since there may be many HIDs associated with a single stream.

hop
A "hop" refers to the portion of a stream's path between two neighbor ST agents. It is usually represented by a physical network. However, a multicast hop can connect a single ST agent to several next-hop ST agents.

host agents
Synonym for host ST agents.

host ST agents
Host ST agents are ST agents that provide services to higher layer protocols and applications. The services include methods for sourcing data from and sinking data to the higher layer or application, and methods for requesting and modifying streams.

intermediate agents
Synonym for intermediate ST agents.

intermediate ST agents
Intermediate ST agents are ST agents that can forward ST packets between the networks to which they are attached.

MTU
The abbreviation for Maximum Transmission Unit, which is the maximum packet size in bytes that can be accepted by a given network for transmission. ST agents determine the maximum packet size for a stream so that data written to the stream can be forwarded through the networks without fragmentation.

multi-destination simplex
The topology and data flow of ST streams are described as being multi-destination simplex: all data flowing on the stream originates from a single origin and is passed to one or more destination targets. Only control information, invisible to the application program, ever passes in the upstream direction.

NAccept
NAccept is an integer parameter maintained by each ST agent. It is used to control retransmission of an ACCEPT message. Since an ACCEPT request is relayed by agents back toward the origin, it must be acknowledged by each previous-hop agent. If this ACK is not received within the appropriate timeout interval, the request will be resent up to NAccept times before giving up.

Name
Generally refers to the name of a stream. A stream Name is structured to ensure that it is unique across all hosts: it includes the address of the host where it was generated combined with a unique number generated at that host. A timestamp is added to ensure that the overall Name is unique over all time. (A stream Name has the same format as a Group Name.)

NConnect
NConnect is an integer parameter maintained by each ST agent. It is used to control retransmission of a CONNECT message. A CONNECT request must be acknowledged by each next-hop agent as it is propagated toward the targets. If a HID-ACCEPT, HID-REJECT, or ACK is not received for the CONNECT between any two agents within the appropriate timeout interval, the request will be resent up to NConnect times before giving up.

NDisconnect
NDisconnect is an integer parameter maintained by each ST agent. It is used to control retransmission of a DISCONNECT message. A DISCONNECT request must be acknowledged by each next-hop agent as it is propagated toward the targets. If this ACK is not received for the DISCONNECT between any two agents within the appropriate timeout interval, the request will be resent up to NDisconnect times before giving up.

next protocol identifier
The next protocol identifier is used by a target ST agent to identify to which of several higher layer protocols it should pass data packets it receives the network. Examples of higher layer protocols include the Network Voice Protocol and the Packet Video Protocol. These higher layer protocols will typically perform further demultiplexing among multiple application processes as part of their protocol processing activities.

next-hop
Synonym for next-hop ST agent.

next-hop ST agent
For each origin or intermediate ST agent managing a stream there are a set of next-hop ST agents. The intermediate agent forwards each data packet it receives to all the next-hop ST agents, which in turn forward the data toward the target host agent (if the particular next-hop agent is another intermediate agent) or to the next higher protocol layer at the target (if the particular next-hop agent is a host agent).

NextPcol
NextPcol is a field in each Target of the CONNECT message used to convey the next protocol identifier. See definition of next protocol identifier above for more details.

NHIDAbort
NHIDAbort is an integer parameter maintained by each ST agent. It is the number of unacceptable HID proposals before an ST agent aborts the HID negotiation process.

NHIDAck
NHIDAck is an integer parameter maintained by each ST agent. It is used to control retransmission of HID-CHANGE-REQUEST messages. HID-CHANGE-REQUEST is sent by an ST agent to the previous-hop ST agent to request that the HID in use between those agents be changed. The previous-hop acknowledges the HID-CHANGE-REQUEST message by sending a HID-CHANGE message. If the HID-CHANGE is not received within the appropriate timeout interval, the request will be resent up to NHIDAck times before giving up.

NHIDChange
NHIDChange is an integer parameter maintained by each ST agent. It is used to control retransmission of the HID-CHANGE message. A HID-CHANGE message must be acknowledged by the next-hop agent. If this ACK is not received within the appropriate timeout interval, the request will be resent up to NHIDChange times before giving up.

NRefuse
NRefuse is an integer parameter maintained by each ST agent. It is used to control retransmission of a REFUSE message. As a REFUSE request is relayed by agents back toward the origin, it must be acknowledged by each previous-hop agent. If this ACK is not received within the appropriate timeout interval, the request will be resent up to NRefuse times before giving up.

NRetryRoute
NRetryRoute is an integer parameter maintained by each ST agent. It is used to control route exploration. When an agent receives a REFUSE message whose ReasonCode indicates that the originally selected route is not acceptable, the agent should attempt to find an alternate route to the target. If the agent has not found a viable route after a maximum of NRetryRoute choices, it should give up and notify the previous-hop or application that it cannot find an acceptable path to the target.

origin
The origin of a stream is the host agent where an application or higher level protocol originally requested that the stream be created. The origin specifies the data to be sent through the stream.

parameter
Parameters are additional values that may be included in control messages. Parameters are often optional. They are distinguished from fields, which are always present.

participants
Participants are the end-users of a stream.

PDU
Abbreviation for Protocol Data Unit, defined below.

peer
The term peer is used to refer to entities at the same protocol layer. It is used here to identify instances of an application or protocol layer above ST. For example, data is passed through a stream from an originating peer process to its target peers.

previous-hop
Synonym for previous-hop ST agent.

previous-hop ST agent
The origin or intermediate agent from which an ST agent receives its data.

protocol data unit
A protocol data unit (PDU) is the unit of data passed to a protocol layer by the next higher layer protocol or user. It consists of control information and possibly user data.

RecoveryTimeout
RecoveryTimeout is specified in the FlowSpec of each stream. The minimum of these values over all streams between a pair of adjacent agents determines how often those agents must send HELLO messages to each other in order to ensure that failure of one of the agents will be detected quickly enough to meet the guarantee implied by the FlowSpec.

Restarted bit
The Restarted bit is part of the HELLO message. When set, it indicates that the sending agent was restarted recently (within the last HelloTimerHoldDown seconds).

round-trip time
The round-trip-time is the time it takes a message to be sent, delivered, processed, and the acknowledgment received. It includes both network and processing delays.

RTT
Abbreviation for round-trip-time.

RVLId
Abbreviation for Receiver's Virtual Link Identifier. It uniquely identifies to the receiver the virtual link, and this stream, used to send it a message. See definition for Virtual Link Identifier below.

SAP
Abbreviation for Service Access Point.

SCMP
Abbreviation for ST Control Message Protocol, defined below.

Service Access Point
A point where a protocol service provider makes available the services it offers to a next higher layer protocol or user.

setup phase
Before data can be transmitted through a stream, the ST agents must distribute state information about the stream to all agents along the path(s) to the target(s). This is the setup phase. The setup phase ends when all the ACCEPT and REFUSE messages sent by the targets have been delivered to the origin. At this point, the data transfer phase begins and data can be sent. Requests to modify the stream can be issued after the setup phase has ended, i.e., during the data transfer phase without disrupting the flow of data.

ST agent
An ST agent is an entity that implements the ST Protocol.

ST Control Message Protocol
The ST Control Message Protocol is the subset of the overall ST Protocol responsible for creation, modification, maintenance, and tear down of a stream. It also includes support for event notification and status monitoring.

stream
A stream is the basic object managed by the ST Protocol for transmission of data. A stream has one origin where data are generated and one or more targets where the data are received for processing. A flow specification, provided by the origin and negotiated among the origin, intermediate, and target ST agents, identifies the requirements of the application and the guarantees that can be assured by the ST agents.

subsets
Subsets of the ST Protocol are permitted, as defined in various sections of this specification. Subsets are defined to allow simplified implementations that can still effectively interoperate with more complete implementations without causing disruption.

SVLId
Abbreviation for Sender's Virtual Link Identifier. It uniquely identifies to the receiver the virtual link identifier that should be placed into the RVLId field of all replies sent over the virtual link for a given stream. See definition for Virtual Link Identifier below.

target
An ST target is the destination where data supplied by the origin will be delivered for higher layer protocol or application processing.

tear down
The tear down phase of a stream begins when the origin indicates that it has no further data to send and the ST agents through which the stream passes should dismantle the stream and release its resources.

ToAccept
ToAccept is a timeout in seconds maintained by each ST agent. It sets the retransmission interval for ACCEPT messages.

ToConnect
ToConnect is a timeout in seconds maintained by each ST agent. It sets the retransmission interval a CONNECT messages.

ToDisconnect
ToDisconnect is a timeout in seconds maintained by each ST agent. It sets the retransmission interval for DISCONNECT messages.

ToHIDAck
ToHIDAck is a timeout in seconds maintained by each ST agent. It sets the retransmission interval for HID-CHANGE-REQUEST messages.

ToHIDChange
ToHIDChange is a timeout in seconds maintained by each ST agent. It sets the retransmission interval for HID-CHANGE messages.

ToRefuse
ToRefuse is a timeout in seconds maintained by each ST agent. It sets the retransmission interval for REFUSE messages.

upstream
The direction in a stream from a target toward the origin.

Virtual Link
A virtual link is one edge of the tree describing the path of data flow through a stream. A separate virtual link is assigned to each pair of neighbor ST agents, even when multiple next-hops are be reached through a single network level multicast group. The virtual link allows efficient demultiplexing of ST Control Message PDUs received from a single physical link or network.

Virtual Link Identifier
For each ST Control Message sent, the sender provides its own virtual link identifier and that of the receiver (if known). Either of these identifiers, combined with the address of the corresponding host, can be used to identify uniquely the virtual control link to the agent. However, virtual link identifiers are chosen by the associated agent so that the agent may precisely identify the stream, state machine, and other protocol processing data elements managed by that agent, without regard to the source of the control message. Virtual link identifiers are not negotiated, and do not change during the lifetime of a stream. They are discarded when the stream is torn down.

7. References

[1] Braden, B., Borman, D., and C. Partridge, "Computing the Internet Checksum", RFC 1071, USC/Information Sciences Institute, Cray Research, BBN Laboratories, September 1988.

[2] Braden, R. (ed.), "Requirements for Internet Hosts -- Communication Layers", RFC 1122, USC/Information Sciences Institute, October 1989.

[3] Cheriton, D., "VMTP: Versatile Message Transaction Protocol Specification", RFC 1045, Stanford University, February 1988.

[4] Cohen, D., "A Network Voice Protocol NVP-II", USC/Information Sciences Institute, April 1981.

[5] Cole, E., "PVP - A Packet Video Protocol", W-Note 28, USC/Information Sciences Institute, August 1981.

[6] Deering, S., "Host Extensions for IP Multicasting", RFC 1112, Stanford University, August 1989.

[7] Edmond W., Seo K., Leib M., and C. Topolcic, "The DARPA Wideband Network Dual Bus Protocol", accepted for presentation at ACM SIGCOMM '90, September 24-27, 1990.

[8] Forgie, J., "ST - A Proposed Internet Stream Protocol", IEN 119, M. I. T. Lincoln Laboratory, 7 September 1979.

[9] Jacobs I., Binder R., and E. Hoversten E., "General Purpose Packet Satellite Network", Proc. IEEE, vol 66, pp 1448-1467, November 1978.

[10] Jacobson, V., "Congestion Avoidance and Control", ACM SIGCOMM-88, August 1988.

[11] Karn, P. and C. Partridge, "Round Trip Time Estimation", ACM SIGCOMM-87, August 1987.

[12] Mallory, T., and A. Kullberg, "Incremental Updating of the Internet Checksum", RFC 1141, BBN Communications Corporation, January 1990.

[13] Mills, D., "Network Time Protocol (Version 2) Specification and Implementation", RFC 1119, University of Delaware, September 1989 (Revised February 1990).

[14] Pope, A., "The SIMNET Network and Protocols", BBN Report No. 7102, BBN Systems and Technologies, July 1989.

[15] Postel, J., ed., "Internet Protocol - DARPA Internet Program Protocol Specification", RFC 791, DARPA, September 1981.

[16] Postel, J., ed., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC 793, DARPA, September 1981.

[17] Postel, J., "User Datagram Protocol", RFC 768, USC/Information Sciences Institute, August 1980.

[18] Reynolds, J., Postel, J., "Assigned Numbers", RFC 1060, USC/Information Sciences Institute, March 1990.

[19] SDNS Protocol and Signaling Working Group, SP3 Sub-Group, SDNS Secure Data Network System, Security Protocol 3 (SP3), SDN.301, Rev. 1.5, 1989-05-15.

[20] SDNS Protocol and Signaling Working Group, SP3 Sub-Group, SDNS Secure Data Network System, Security Protocol 3 (SP3) Addendum 1, Cooperating Families, SDN.301.1, Rev. 1.2, 1988-07-12.

8. Security Considerations

See section 3.7.8.

9. Authors' Addresses

Stephen Casner
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695

Phone: (213) 822-1511 x153
EMail: Casner@ISI.Edu

Charles Lynn, Jr.
BBN Systems and Technologies,
a division of Bolt Beranek and Newman Inc.
10 Moulton Street
Cambridge, MA 02138

Phone: (617) 873-3367
EMail: CLynn@BBN.Com

Philippe Park
BBN Systems and Technologies,
a division of Bolt Beranek and Newman Inc.
10 Moulton Street
Cambridge, MA 02138

Phone: (617) 873-2892
EMail: ppark@BBN.COM

Kenneth Schroder
BBN Systems and Technologies,
a division of Bolt Beranek and Newman Inc.
10 Moulton Street
Cambridge, MA 02138

Phone: (617) 873-3167
EMail: Schroder@BBN.Com

Claudio Topolcic
BBN Systems and Technologies,
a division of Bolt Beranek and Newman Inc.
10 Moulton Street
Cambridge, MA 02138

Phone: (617) 873-3874
EMail: Topolcic@BBN.Com

[This page intentionally left blank.]

Appendix 1. Data Notations

The convention in the documentation of Internet Protocols is to express numbers in decimal and to picture data with the most significant octet on the left and the least significant octet on the right.

The order of transmission of the header and data described in this document is resolved to the octet level. Whenever a diagram shows a group of octets, the order of transmission of those octets is the normal order in which they are read in English. For example, in the following diagram the octets are transmitted in the order they are numbered.

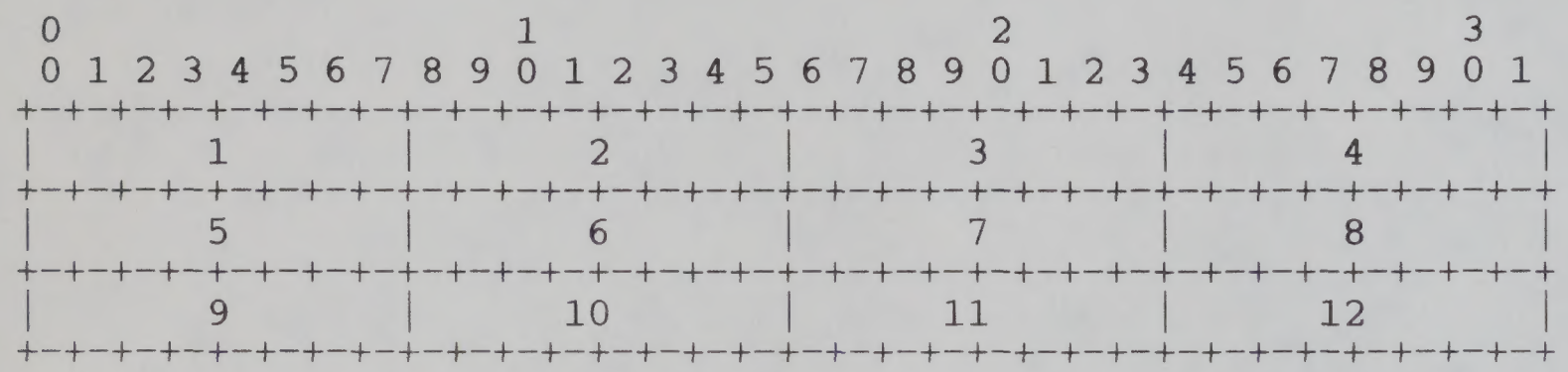


Figure 56. Transmission Order of Bytes

Whenever an octet represents a numeric quantity the left most bit in the diagram is the high order or most significant bit. That is, the bit labeled 0 is the most significant bit. For example, the following diagram represents the value 170 (decimal).

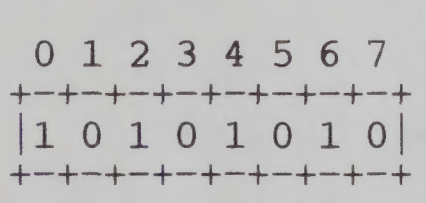


Figure 57. Significance of Bits

Similarly, whenever a multi-octet field represents a numeric quantity the left most bit of the whole field is the most significant bit. When a multi-octet quantity is transmitted the most significant octet is transmitted first.

Fields whose length is fixed and fully illustrated are shown with a vertical bar (|) at the end; fixed fields whose contents are abbreviated are shown with an exclamation point (!); variable fields are shown with colons (:).

Optional parameters are separated from control messages with a blank line. The order of any optional parameters is not meaningful.